A FAST MULTI-PURPOSE CIRCUIT SIMULATOR USING THE LATENCY INSERTION METHOD

BY

PATRICK KUANLYE GOH

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor José E. Schutt-Ainé, Chair
Professor Jennifer T. Bernhard
Professor Andreas C. Cangellaris
Professor Martin D. F. Wong

ABSTRACT

With the increase in the density of interconnects and the complexity of high-speed packages, signal integrity becomes an important aspect in the design of modern devices. Circuit designers are constantly in need of robust circuit simulation methods that are able to capture the complicated electromagnetic behaviors of complex circuits, and do it in a fraction of the time taken by conventional circuit simulators. As a result, there is a constant need for and push toward faster and more accurate circuit simulation techniques.

The latency insertion method (LIM) has recently emerged as an efficient approach for performing fast simulations of very large circuits. By exploiting latencies in the circuit, LIM is able to solve the voltages and the currents in the circuit explicitly at each time step. This results in a computationally efficient algorithm that is able to simulate large circuits significantly faster than traditional matrix inversion-based methods such as SPICE.

In this work, we propose the use of LIM as a multi-purpose circuit simulator. While LIM originated mainly as a means for performing fast transient simulations of high-speed interconnects characterized by RLGC elements, we aim to provide additional derivations of and modifications to LIM in order to formulate a robust circuit simulator that is both fast and accurate.

To all graduate students
striving to make a difference,
no matter how small

ACKNOWLEDGMENTS

"If I have seen a little further it is by standing on the shoulders of giants."

Sir Isaac Newton

It is said that we are like kites in the sky, soaring high not because we can fly, but because we are lifted by the wind, and held at an angle by the string. I have always believed that everything that I have accomplished, and all that I have earned to bring me to where I am today, is earned not by virtue of any distinction on my part, but because I have been blessed to be surrounded by such kind people, who have made me a better person than I really am. In this short page, I would like to express my sincere gratitude, for all the support that I have acquired from everyone around me. Following is a short list of all the individuals that I can remember. Apologies for any that I missed.

My advisor, Prof. José Schutt-Ainé; committee members—Prof. Jennifer Bernhard, Prof. Andreas Cangellaris and Prof. Martin Wong; group members—Dmitri Klokotov, Pavle Milosevic, Si Win, Tom Comberiate and Daniel Chang; and the people at Cadence—Jilin Tan, Ping Liu and Feras Al-Hawari.

Last and certainly not least, I will always be grateful to my parents and my sister for their love, support and understanding as I pursued my dream. I might be a long way from where I started, but I certainly have not forgotten my way home.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

With the advancement of recent technology and the ongoing effort to achieve faster and smaller electronic systems, developers continue to push the envelope in terms of operating frequencies and design densities. Circuits operating in the gigahertz range, with dense and complex three-dimensional interconnect schemes, are common in the industry today. This trend, however, has led to a number of issues in the design and operation of modern circuits. The increase in signal speed leads to an increase in the significance of signal integrity issues such as crosstalk, dispersion, attenuation, reflection, delay and distortion. In addition, the increase in density of interconnects leads to very large circuits which render their analysis more challenging from a runtime perspective. As a result, the simulation of very large networks for signal integrity analysis has become a prominent subject of research in the past few decades [1, 2].

In the field of interconnect simulations, current research includes the work on macromodeling and model-order reduction, which deals with approximating the network transfer functions by a smaller and simplified representation that captures the main behavior of the network over the frequency range of interest, and the research on fast time-domain circuit simulators which deals with either special types of circuits such as transmission lines or power distribution networks or more general circuits provided by an extractor. Some examples of past research on macromodeling and

model-order reduction can be found in [3–17]. In [3–5], the method of characteristics is used to transform the partial differential equations of transmission lines into ordinary differential equations. The method can be applied in both the time and frequency domain and can be extended to handle lossy [5] and multiconductor lines [6]. In [7–11], a least square approximation is used to derive a transfer function representation of the system in the form of a rational function. The resulting poles and residues are then easily converted into a set of ordinary differential equations in the time domain. More recent work in this area has focused on ensuring the passivity of the resulting approximation [12–16]. In addition, model-order reduction techniques such as the moment matching technique have been applied to reduce the number of poles by capturing only the dominant poles of the system [17]. Research efforts on fast time-domain circuit simulators are typically derived from the finite-difference time-domain (FDTD) algorithm which is then applied to transmission lines [18, 19]. The latency insertion method (LIM) [20], which is the subject of this research, initially arose as a fast time-domain simulation method for the transient simulations of interconnects.

1.2 Objective

The simulation of very large networks is a common and prevailing problem in the design of integrated circuits. In the field of computer-aided design, large networks are characterized by a netlist which contains a large number of nodes and circuit elements. Simulations of such circuits are typically very time-consuming and suffer from large memory requirements which can be impractical in certain scenarios. For this reason, there has been a constant push towards faster circuit simulation methods that are able to handle large circuits in a fraction of the time of conventional circuit simulators.

The Simulation Program with Integrated Circuit Emphasis (SPICE) [21] is the current industry standard for general-purpose circuit simulators. SPICE utilizes the modified nodal analysis (MNA) method which solves a set of linear equations constructed from the circuit. The main reasons for the popularity of SPICE are: (1) its general-purpose nature, in that it is able to simulate almost all types of circuit elements, (2) its accuracy, and (3) its open source nature and long standing in the industry. However, SPICE is not without its weaknesses. Its reliance on forming large matrix systems results in large memory requirements and the subsequent matrix inversion process is inefficient in terms of runtime. Also, when nonlinear devices are present, SPICE relies on the Newton-Raphson iterative process which suffers from convergent problems and further speed issues due to the expensive LU decomposition process. As a result, there have been many commercial tools which aim at improving the many facets of SPICE for faster circuit simulations; these tools include SPECTRE [22], ELDO [23] and Analog FastSPICE [24].

In this work, we propose the use of the latency insertion method as a multi-purpose circuit simulator. The aim is to provide LIM with the necessary derivation and modifications in order to achieve a circuit simulator that is:

1. Able to handle various circuit elements such as resistors, inductors, capacitors, sources, dependent sources, transistors and even blackbox models.
2. As accurate as SPICE.
3. Faster than SPICE (ideally a linear runtime with respect to the number of nodes).
4. Able to perform advanced speed-up methods such as circuit partitioning and multi-rate simulations.

1.3 Organization

The remainder of the dissertation is organized as follows. Chapter 2 presents some background information on the latency insertion method. This includes the basic LIM derivation and recent advancements in LIM such as Block-LIM, its stability analysis and the recently developed formulation for dependent sources. The chapter concludes with some examples and comparison with existing methods.

Chapter 3 presents the newly developed partitioned latency insertion method which utilizes a generalized stability criterion. By partitioning circuits with multiple latencies and utilizing a different time step for each partition, further speed-up is obtained in the LIM. The time step for each partition is selected based on its maximum stable time step using the stability criterion. A numerical example is presented to demonstrate the method and its improvement over the traditional LIM.

Chapter 4 illustrates how blackbox macromodeling techniques can be used in conjunction with the LIM. The chapter starts with an overview of two different blackbox macromodeling techniques, which are model-order reduction via vector fitting and a fast convolution method. Related issues with each method such as passivity and causality enforcements are discussed in detail. Next, the integration process of performing LIM simulations with blackbox macromodels is illustrated and examples presented. The chapter concludes with a comparative study of the two methods.

In Chapter 5, the recently developed CMOS simulation technique using the LIM is presented. The method is extended to perform multi-rate simulations, and numerical examples, which show the accuracy and computational efficiency of the new method, are given.

Chapter 6 explores the subject of analog circuit simulations using the LIM. Specifically, LIM is applied to the simulations of phase-locked loops (PLLs). Two different approaches are shown, first on a behavioral level by using the PLL governing equa-

tions, and next, on a full transistor level. Comparisons between the two methods and with existing commercial circuit simulators are depicted.

Finally, Chapter 7 presents a short conclusion and proffers some future work on the subject.

CHAPTER 2

BACKGROUND

2.1 Basic LIM Formulation

In this section, the formulation of the basic LIM is presented. LIM can be applied to any arbitrary network, where it is assumed that through the use of Thévenin and Norton transformations, the branches and nodes of the circuit can be described by a general topology. Each node is represented by a parallel combination of a current source, a conductance, and a capacitor to ground. The connection between two different nodes forms a branch and it is represented by a series combination of a voltage source, a resistor and an inductor. Fig. 2.1 shows a node i with k branches connected to it, while Fig. 2.2 shows a branch connecting nodes i and j. The voltage at node i is defined as V_i while the current flowing from node i to j is defined as I_{ij}. In order to solve for the voltages and currents in the circuit, LIM discretizes the time variable whereby the voltages and currents are collated in half time steps. Specifically, the voltages are solved at half time steps while the currents are solved at full time steps. From Fig. 2.1, writing Kirchhoff's current law (KCL) at node i yields

$$C_i \left(\frac{V_i^{n+1/2} - V_i^{n-1/2}}{\Delta t} \right) + G_i V_i^{n-1/2} - H_i^n = -\sum_{k=1}^{M_i} I_{ik}^n \tag{2.1}$$

where the superscript n is the index of the current time step, Δt is the time step and M_i is the number of branches connected to node i. Solving for the unknown voltage

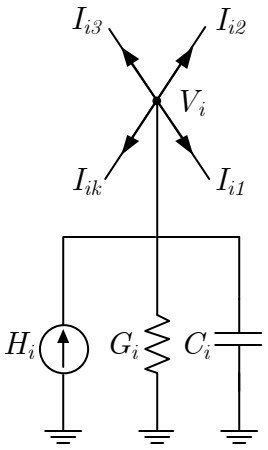


Figure 2.1: Node topology.

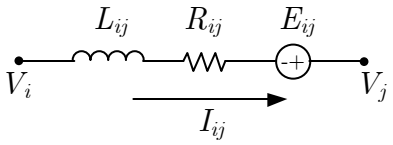


Figure 2.2: Branch topology.

yields

$$V_i^{n+1/2} = V_i^{n-1/2} + \frac{\Delta t}{C_i}\left(-\sum_{k=1}^{M_i} I_{ik}^n - G_i V_i^{n-1/2} + H_i^n\right) \tag{2.2}$$

for $i = 1, 2, ..., N_n$, where N_n is the number of nodes in the circuit.

From Fig. 2.2, writing Kirchhoff's voltage law (KVL) at branch ij yields

$$V_i^{n+1/2} - V_j^{n+1/2} = L_{ij}\left(\frac{I_{ij}^{n+1} - I_{ij}^n}{\Delta t}\right) + R_{ij} I_{ij}^n - E_{ij}^{n+1/2}. \tag{2.3}$$

Solving for the unknown current yields

$$I_{ij}^{n+1} = I_{ij}^n + \frac{\Delta t}{L_{ij}}\left(V_i^{n+1/2} - V_j^{n+1/2} - R_{ij} I_{ij}^n + E_{ij}^{n+1/2}\right). \tag{2.4}$$

The computation of the node voltages and the branch currents are alternated as time progresses in a leapfrog manner. In this aspect, LIM is similar to Yee's algorithm for the solution of Maxwell's equations in the finite-difference time-domain (FDTD) method [25]. It is clear that the LIM algorithm relies on the latencies in the network to perform the leapfrog time stepping formulation. Thus, at every node, a capacitor to ground has to be present. If it is not, a small fictitious capacitor is inserted.

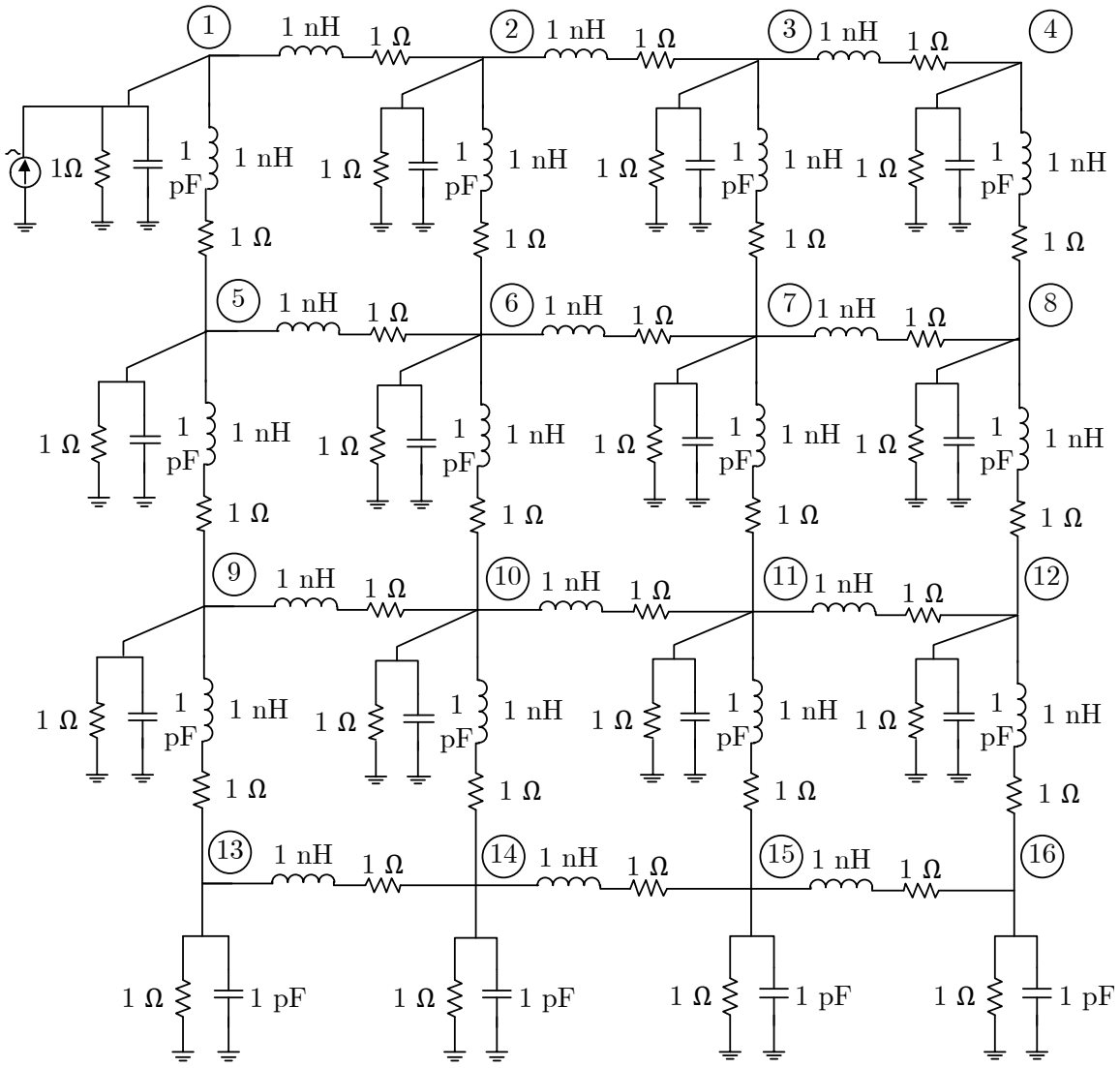


Figure 2.3: RLGC grid.

Similarly, small fictitious inductors are inserted into branches without latencies. As with the traditional FDTD method [25], LIM is only conditionally stable. In other words, there is an upper bound on the time step that will result in a numerically stable solution to (2.2) and (2.4). This will be studied in the following section.

Before we proceed, we present a simple example to illustrate the benefit of LIM over conventional SPICE-like circuit simulators. Consider the 4×4 RLGC grid-type circuit shown in Fig. 2.3, which represents a general interconnect topology. The circuit is

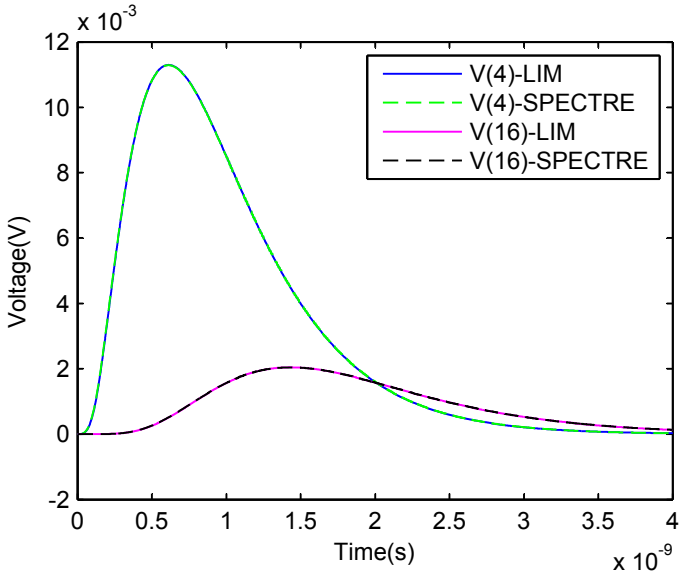


Figure 2.4: Outputs at nodes 4 and 16 of Fig 2.3 for both methods.

driven at node 1 by a current source with a trapezoidal pulse with rise and fall times of 10 ps, a pulse width of 100 ps and a magnitude of 6 A. LIM is used to simulate the circuit with a time step equal to $(risetime)/10 = 1$ ps in order to accurately capture behaviors at the rising and falling edges of the input. The simulation time is 4 ns. For comparison, the same circuit is also simulated in SPECTRE [22], a commercial simulation tool from Cadence Design Systems Inc., which utilizes the SPICE-like modified nodal analysis (MNA) method. In both cases, the simulations are performed on a Linux server with Intel Xeon 3.16 GHz processors and 32 GB of RAM. The output at nodes 4 and 16 are compared and shown in Fig. 2.4. Comparable accuracy is observed between the two methods. Next, larger grids are constructed and simulated and the runtime for each method is recorded in Table 2.1 and shown in Fig. 2.5. We see that LIM exhibits a linear numerical complexity with respect to the number of nodes and clearly outperforms SPECTRE in this example.

Before concluding this section, we present two alternate formulations for the LIM algorithm. In (2.1) and (2.3), the terms $G_i V_i^{n-1/2}$ and $R_{ij} I_{ij}^n$ are used for the con-

Table 2.1: Comparison of runtime for LIM and SPECTRE.

Circuit Size	# nodes (LIM)	# nodes (SPECTRE)†	SPECTRE (s)	LIM (s)
20 × 20	400	1160	0.15	1.53
40 × 40	1600	4720	2.14	6.25
60 × 60	3600	10680	25.73	14.23
80 × 80	6400	19040	140.59	25.71
100 × 100	10000	29800	445.77	40.64
120 × 120	14400	42960	945.00	62.89

† In SPECTRE and other SPICE-like simulators, the connection between the resistor and inductor in the branch is treated as an extra node.

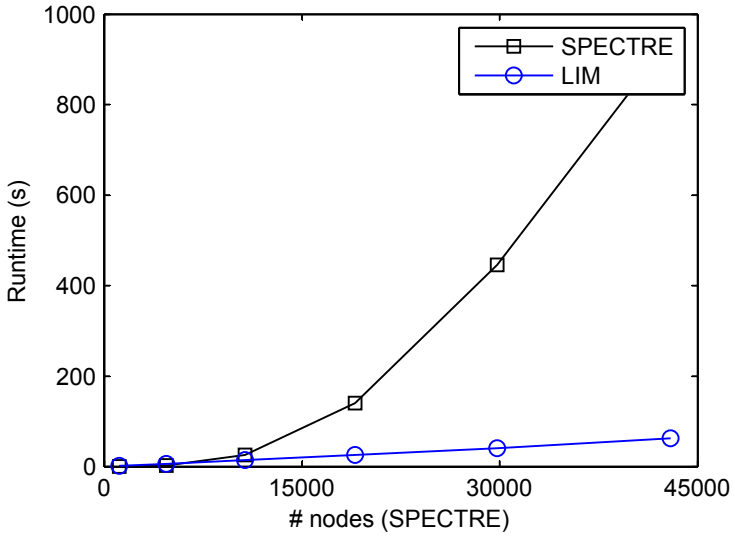


Figure 2.5: Comparison of runtime for LIM and SPECTRE.

ductance and resistance terms respectively. This is the fully explicit formulation. An alternate formulation is possible by substituting the terms $G_i V_i^{n+1/2}$ and $R_{ij} I_{ij}^{n+1}$ in place of the aforementioned conductance and resistance terms. This is the fully implicit formulation where the updating equations of (2.2) and (2.4) are now given by the following two equations:

$$V_i^{n+1/2} = \left(\frac{C_i}{\Delta t} + G_i \right)^{-1} \cdot \left(\frac{C_i}{\Delta t} V_i^{n-1/2} - \sum_{k=1}^{M_i} I_{ik}^n + H_i^n \right) \tag{2.5}$$

$$I_{ij}^{n+1} = \left(\frac{L_{ij}}{\Delta t} + R_{ij}\right)^{-1} \cdot \left(\frac{L_{ij}}{\Delta t} I_{ij}^{n} + V_i^{n+1/2} - V_j^{n+1/2} + E_{ij}^{n+1/2}\right). \tag{2.6}$$

A third alternate formulation is possible by substituting the conductance and resistance terms as $G_i \left(V_i^{n+1/2} + V_i^{n-1/2}\right)/2$ and $R_{ij}\left(I_{ij}^{n+1} + I_{ij}^{n}\right)/2$ respectively. This is the semi-implicit formulation which will be utilized in the Block-LIM formulation in the next section.

2.2 Advancement in LIM

In this section, the recent advancements in LIM are presented. First the vector-matrix LIM or Block-LIM is presented. Next a stability analysis to determine the maximum stable time step for a LIM simulation is performed using the newly introduced Block-LIM formulation. This leads to the definition of the amplification matrix. Finally, the modifications to include dependent sources are described, along with the resulting modifications to the amplification matrix.

2.2.1 Block-LIM

In this section, the formulation of the vector-matrix version of LIM or Block-LIM [26] is presented. From (2.1), we may write the semi-implicit formulation as

$$C_i \left(\frac{V_i^{n+1/2} - V_i^{n-1/2}}{\Delta t}\right) + G_i \left(\frac{V_i^{n+1/2} + V_i^{n-1/2}}{2}\right) - H_i^n = -\sum_{k=1}^{M_i} I_{ik}^n. \tag{2.7}$$

Equation (2.7) can then be written in a vector-matrix formulation as

$$\mathbf{C}\left(\frac{\mathbf{v}^{n+1/2} - \mathbf{v}^{n-1/2}}{\Delta t}\right) + \frac{1}{2}\mathbf{G}\left(\mathbf{v}^{n+1/2} + \mathbf{v}^{n-1/2}\right) - \mathbf{h}^n = -\mathbf{M}\mathbf{i}^n \tag{2.8}$$

where $\mathbf{v}$ is the node voltage vector of dimension N_n, $\mathbf{i}$ is the branch current vector of dimension N_b, $\mathbf{C}$ and $\mathbf{G}$ are diagonal matrices respectively of dimensions $N_n \times N_n$,

with the values of the capacitors and conductances at each node on the main diagonal, $\mathbf{h}$ is a vector of dimension N_n containing all the current sources at the nodes and $\mathbf{M}$ is the $N_n \times N_b$ incidence matrix defined as follows:

$$\begin{aligned} M_{qp} &= 1 \quad \textit{if branch } p \textit{ is incident at node } q \textit{ and the} \\ &\quad\ \ \textit{current flows away from node } q \\ M_{qp} &= -1 \quad \textit{if branch } p \textit{ is incident at node } q \textit{ and the} \\ &\quad\ \ \textit{current flows into node } q \\ M_{qp} &= 0 \quad \textit{if branch } p \textit{ is not incident at node } q \end{aligned}$$

Solving (2.8) for $\mathbf{v}^{n+1/2}$ yields

$$\mathbf{v}^{n+1/2} = \left(\frac{\mathbf{C}}{\Delta t} + \frac{\mathbf{G}}{2}\right)^{-1} \left[\left(\frac{\mathbf{C}}{\Delta t} - \frac{\mathbf{G}}{2}\right)\mathbf{v}^{n-1/2} + \mathbf{h}^n - \mathbf{M}\mathbf{i}^n\right]. \tag{2.9}$$

Similarly, we may write the semi-implicit formulation of (2.3) in vector-matrix form as

$$\mathbf{M}^T\mathbf{v}^{n+1/2} = \frac{\mathbf{L}}{\Delta t}\left(\mathbf{i}^{n+1} - \mathbf{i}^n\right) + \frac{\mathbf{R}}{2}\left(\mathbf{i}^{n+1} + \mathbf{i}^n\right) - \mathbf{e}^{n+1/2} \tag{2.10}$$

where $\mathbf{L}$ and $\mathbf{R}$ are diagonal matrices respectively of dimensions $N_b \times N_b$, with the values of the inductances and resistances at each branch on the main diagonal, and $\mathbf{e}$ is a vector of dimension N_b containing all the voltage sources at the branches. Solving (2.10) for $\mathbf{i}^{n+1}$ yields

$$\mathbf{i}^{n+1} = \left(\frac{\mathbf{L}}{\Delta t} + \frac{\mathbf{R}}{2}\right)^{-1} \left[\left(\frac{\mathbf{L}}{\Delta t} - \frac{\mathbf{R}}{2}\right)\mathbf{i}^n + \mathbf{e}^{n+1/2} + \mathbf{M}^T\mathbf{v}^{n+1/2}\right]. \tag{2.11}$$

Equations (2.9) and (2.11) can then be used in place of (2.2) and (2.4) as the update equations to calculate the voltage and currents at each time step.

2.2.2 Stability Analysis

The advantage of the vector-matrix formulation lies in its ability to accurately predict if a time step will be stable. To see this, we return to (2.9) and (2.11) and expand them to get

$$\mathbf{v}^{n+1/2} = \mathbf{P}_+\mathbf{P}_-\mathbf{v}^{n-1/2} - \mathbf{P}_+\mathbf{M}\mathbf{i}^n + \mathbf{P}_+\mathbf{h}^n \tag{2.12}$$

$$\mathbf{i}^{n+1} = \mathbf{Q}_+\mathbf{Q}_-\mathbf{i}^n + \mathbf{Q}_+\mathbf{M}^T\mathbf{v}^{n+1/2} + \mathbf{Q}_+\mathbf{e}^{n+1/2} \tag{2.13}$$

where we have made the definitions

$$\mathbf{P}_+ = \left(\frac{\mathbf{C}}{\Delta t} + \frac{\mathbf{G}}{2}\right)^{-1} \qquad \mathbf{P}_- = \left(\frac{\mathbf{C}}{\Delta t} - \frac{\mathbf{G}}{2}\right) \tag{2.14}$$

$$\mathbf{Q}_+ = \left(\frac{\mathbf{L}}{\Delta t} + \frac{\mathbf{R}}{2}\right)^{-1} \qquad \mathbf{Q}_- = \left(\frac{\mathbf{L}}{\Delta t} - \frac{\mathbf{R}}{2}\right). \tag{2.15}$$

Substituting (2.12) into (2.13) and rearranging the terms, we obtain

$$\begin{aligned} \mathbf{i}^{n+1} \;=\; & \mathbf{Q}_+\mathbf{M}^T\mathbf{P}_+\mathbf{P}_-\mathbf{v}^{n-1/2} + \left(\mathbf{Q}_+\mathbf{Q}_- - \mathbf{Q}_+\mathbf{M}^T\mathbf{P}_+\mathbf{M}\right)\mathbf{i}^n \\ & + \mathbf{Q}_+\mathbf{e}^{n+1/2} + \mathbf{Q}_+\mathbf{M}^T\mathbf{P}_+\mathbf{h}^n. \end{aligned} \tag{2.16}$$

Equations (2.12) and (2.16) can then be grouped together to obtain

$$\begin{aligned} \begin{bmatrix} \mathbf{v}^{n+1/2} \\ \mathbf{i}^{n+1} \end{bmatrix} \;=\; & \begin{bmatrix} \mathbf{P}_+\mathbf{P}_- & -\mathbf{P}_+\mathbf{M} \\ \mathbf{Q}_+\mathbf{M}^T\mathbf{P}_+\mathbf{P}_- & \mathbf{Q}_+\mathbf{Q}_- - \mathbf{Q}_+\mathbf{M}^T\mathbf{P}_+\mathbf{M} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{n-1/2} \\ \mathbf{i}^n \end{bmatrix} \\ & + \begin{bmatrix} 0 & \mathbf{P}_+ \\ \mathbf{Q}_+ & \mathbf{Q}_+\mathbf{M}^T\mathbf{P}_+ \end{bmatrix} \begin{bmatrix} \mathbf{e}^{n+1/2} \\ \mathbf{h}^n \end{bmatrix}. \end{aligned} \tag{2.17}$$

Equation (2.17) defines a discrete linear time-invariant system (DLTI) in the form of

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t). \tag{2.18}$$

Theorem 1: The DLTI given in (2.18) is asymptotically stable if and only if all the eigenvalues of $\mathbf{A}$ have magnitude strictly smaller than 1. The reader is referred to [27] for a proof of this theorem.

Comparing (2.17) and (2.18), we define the matrix $\mathbf{A}$ as

$$\mathbf{A} = \begin{bmatrix} \mathbf{P}_+\mathbf{P}_- & -\mathbf{P}_+\mathbf{M} \\ \mathbf{Q}_+\mathbf{M}^T\mathbf{P}_+\mathbf{P}_- & \mathbf{Q}_+\mathbf{Q}_- - \mathbf{Q}_+\mathbf{M}^T\mathbf{P}_+\mathbf{M} \end{bmatrix} \tag{2.19}$$

and call it the amplification matrix since in the absence of input, the voltages and the currents in the circuit will be amplified by the matrix $\mathbf{A}$ at each time step. From Theorem 1, we see that *all the eigenvalues of the amplification matrix defined in* (2.19) *must have magnitude strictly smaller than 1* for the simulation to be stable. Thus, we can use the amplification matrix to predict the stability of a time step Δt.

2.2.3 Dependent Sources

In this section, we develop the voltage and current update equations in the presence of dependent sources, and the resulting modification to the amplification matrix [28–31].

Fig. 2.6 shows the node topology with a voltage-controlled current source (VCCS) and a current-controlled current source (CCCS) connected to it. Writing the KCL at

the node, in semi-implicit form, gives

$$\begin{aligned} C_i\left(\frac{V_i^{n+1/2}-V_i^{n-1/2}}{\Delta t}\right)+G_i\left(\frac{V_i^{n+1/2}+V_i^{n-1/2}}{2}\right)-H_i^n \\ -B_{ik}\left(\frac{V_k^{n+1/2}+V_k^{n-1/2}}{2}\right)-S_{ip}I_p^n=-\sum_{k=1}^{M_i}I_{ik}^n \end{aligned} \tag{2.20}$$

where B_{ik} is the coefficient of the VCCS at node i due to node k and S_{ip} is the coefficient of the CCCS at node i due to branch p. Equation (2.20) can then be written in vector-matrix form as

$$\begin{aligned} \mathbf{C}\left(\frac{\mathbf{v}^{n+1/2}-\mathbf{v}^{n-1/2}}{\Delta t}\right)+\frac{1}{2}\mathbf{G}\left(\mathbf{v}^{n+1/2}+\mathbf{v}^{n-1/2}\right)-\mathbf{h}^n \\ -\frac{1}{2}\mathbf{B}\left(\mathbf{v}^{n+1/2}+\mathbf{v}^{n-1/2}\right)-\mathbf{S}\mathbf{i}^n=-\mathbf{M}\mathbf{i}^n \end{aligned} \tag{2.21}$$

which can be rearranged to read

$$\mathbf{C}\left(\frac{\mathbf{v}^{n+1/2}-\mathbf{v}^{n-1/2}}{\Delta t}\right)+\frac{1}{2}\mathbf{G}'\left(\mathbf{v}^{n+1/2}+\mathbf{v}^{n-1/2}\right)-\mathbf{h}^n=-\mathbf{M}'\mathbf{i}^n \tag{2.22}$$

where

$$\mathbf{G}'=\mathbf{G}-\mathbf{B} \quad \text{and} \quad \mathbf{M}'=\mathbf{M}-\mathbf{S}. \tag{2.23}$$

Solving (2.22) for $\mathbf{v}^{n+1/2}$ yields

$$\mathbf{v}^{n+1/2}=\left(\frac{\mathbf{C}}{\Delta t}+\frac{\mathbf{G}'}{2}\right)^{-1}\left[\left(\frac{\mathbf{C}}{\Delta t}-\frac{\mathbf{G}'}{2}\right)\mathbf{v}^{n-1/2}+\mathbf{h}^n-\mathbf{M}'\mathbf{i}^n\right] \tag{2.24}$$

which is the voltage update equation in the presence of dependent sources.

Fig. 2.7 shows the branch topology with a voltage-controlled voltage source (VCVS) and a current-controlled voltage source (CCVS) connected to it. KVL at the branch,

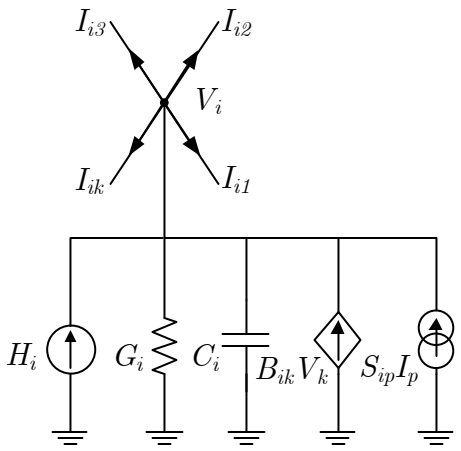


Figure 2.6: Node with dependent sources.

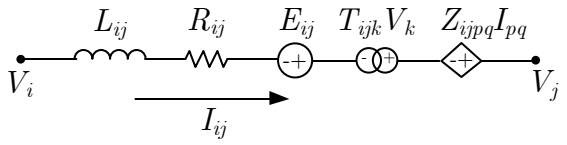


Figure 2.7: Branch with dependent sources.

in semi-implicit form, gives

$$\begin{aligned} V_i^{n+1/2} - V_j^{n+1/2} &= L_{ij}\left(\tfrac{I_{ij}^{n+1}-I_{ij}^{n}}{\Delta t}\right) + R_{ij}\left(\tfrac{I_{ij}^{n+1}+I_{ij}^{n}}{2}\right) - E_{ij}^{n+1/2} \\ &\quad -T_{ijk}V_k^{n+1/2} - Z_{ijpq}\left(\frac{I_{pq}^{n+1}+I_{pq}^{n}}{2}\right) \end{aligned} \tag{2.25}$$

where T_{ijk} is the coefficient of the VCVS at branch ij due to node k and Z_{ijpq} is the coefficient of the CCVS at branch ij due to branch pq. Writing (2.25) in vector-matrix form and rearranging the terms, we obtain

$$\mathbf{M}^{T'}\mathbf{v}^{n+1/2} = \frac{\mathbf{L}}{\Delta t}\left(\mathbf{i}^{n+1} - \mathbf{i}^{n}\right) + \frac{\mathbf{R}'}{2}\left(\mathbf{i}^{n+1} + \mathbf{i}^{n}\right) - \mathbf{e}^{n+1/2} \tag{2.26}$$

where

$$\mathbf{M}^{T'} = \mathbf{M}^{T} + \mathbf{T} \quad \text{and} \quad \mathbf{R}' = \mathbf{R} - \mathbf{Z}. \tag{2.27}$$

Solving (2.26) for i^{n+1} yields

$$\mathbf{i}^{n+1} = \left(\frac{\mathbf{L}}{\Delta t} + \frac{\mathbf{R}'}{2}\right)^{-1}\left[\left(\frac{\mathbf{L}}{\Delta t} - \frac{\mathbf{R}'}{2}\right)\mathbf{i}^{n} + \mathbf{e}^{n+1/2} + \mathbf{M}^{T'}\mathbf{v}^{n+1/2}\right]. \tag{2.28}$$

Equations (2.24) and (2.28) then give the new update equations for circuits with dependent sources. Note that in the absence of dependent sources, all the $\mathbf{G}'$, $\mathbf{M}'$, $\mathbf{M}^{T'}$ and $\mathbf{R}'$ will converge to $\mathbf{G}$, $\mathbf{M}$, $\mathbf{M}^T$ and $\mathbf{R}$, and (2.24) and (2.28) will converge to (2.9) and (2.11) as expected.

In order to analyze the stability of a time step in the presence of dependent sources, we proceed as in the previous section, to obtain the new amplification matrix $\mathbf{A}'$, where we now have

$$\mathbf{A}' = \begin{bmatrix} \mathbf{P}_+{}'\mathbf{P}_-{}' & -\mathbf{P}_+{}'\mathbf{M}' \\ \mathbf{Q}_+{}'\mathbf{M}^{T'}\mathbf{P}_+{}'\mathbf{P}_-{}' & \mathbf{Q}_+{}'\mathbf{Q}_-{}' - \mathbf{Q}_+{}'\mathbf{M}^{T'}\mathbf{P}_+{}'\mathbf{M}' \end{bmatrix} \tag{2.29}$$

where

$$\mathbf{P}_+{}' = \left(\frac{\mathbf{C}}{\Delta t} + \frac{\mathbf{G}'}{2}\right)^{-1} \qquad \mathbf{P}_-{}' = \left(\frac{\mathbf{C}}{\Delta t} - \frac{\mathbf{G}'}{2}\right) \tag{2.30}$$

$$\mathbf{Q}_+{}' = \left(\frac{\mathbf{L}}{\Delta t} + \frac{\mathbf{R}'}{2}\right)^{-1} \qquad \mathbf{Q}_-{}' = \left(\frac{\mathbf{L}}{\Delta t} - \frac{\mathbf{R}'}{2}\right). \tag{2.31}$$

From Theorem 1, we see that *all the eigenvalues of the amplification matrix* $\mathbf{A}'$ *defined in* (2.29) *must have magnitude strictly smaller than 1* for the simulation with dependent sources to be stable. This is written compactly as follows:

$$|\lambda_i\left(\mathbf{A}'(\Delta t)\right)| < 1 \qquad i = 1, 2, \ldots, N_n + N_b. \tag{2.32}$$

Thus, we can use the new amplification matrix $\mathbf{A}'$ to predict the stability of a time step Δt in the presence of dependent sources.

2.2.4 Example

In this section, the methods presented in the previous sections are applied to perform a LIM simulation in the presence of dependent sources. The developed stability criteria will also be verified.

Consider the circuit shown in Fig. 2.8, which contains four dependent sources (VCCS, CCCS, VCVS and CCVS). It is assumed that all the branches and the nodes in the circuit have inherent latencies as shown in the figure such that no fictitious elements have to be inserted. The input is a current source with a single trapezoidal pulse of rise and fall times equal to 1 ns and a pulse width of 4 ns. The maximum amplitude is 0.02 A. In order to validate Theorem 1 and the subsequent result in (2.32), a sweep of the eigenvalues of the amplification matrix $\mathbf{A}'$ is performed and the maximum time step for stability is determined from where the magnitude of the maximum eigenvalue equals 1. This is shown in Fig. 2.9. Note that in practice, this process can be time-consuming, especially when the circuit under consideration is large. In that case, the circuit can be partitioned into multiple segments and the different partitions can be simulated with different time steps. This will be explained in the next chapter. Also, a more effective search algorithm can be employed to determine the maximum time step.

From Fig. 2.9, the maximum time step is determined to be $\Delta t_{max} < 6.864 \times 10^{-11}$ s. We then perform two LIM transient simulations, first using a time step slightly smaller than the maximum time step and then using a time step slightly larger than the maximum time step. In order to validate the method, the same circuit is also simulated in SPECTRE [22], a commercial simulation tool from Cadence Design Systems Inc., which utilizes the SPICE-like modified nodal analysis (MNA) method. Plots of the resulting waveforms at the input (node 1) and output (node 4) are shown in Fig. 2.10. We see that the simulation using the properly chosen time step

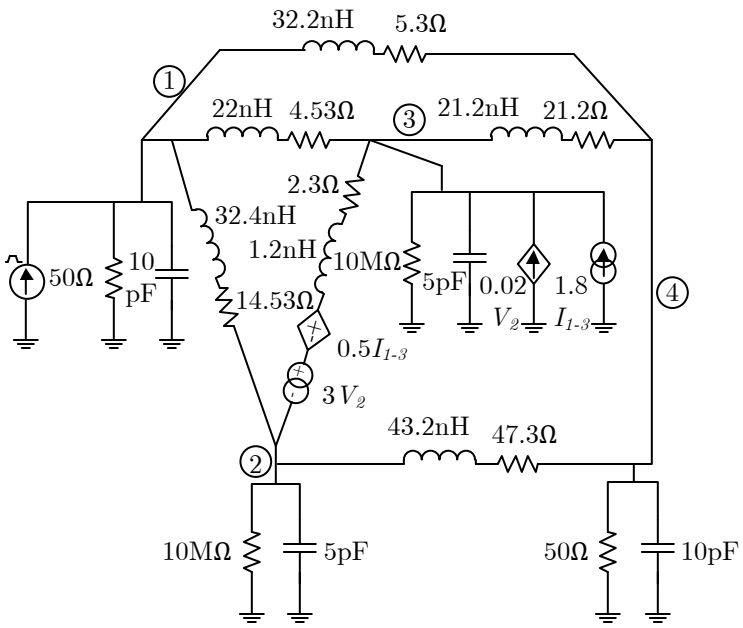


Figure 2.8: Example circuit with dependent sources.

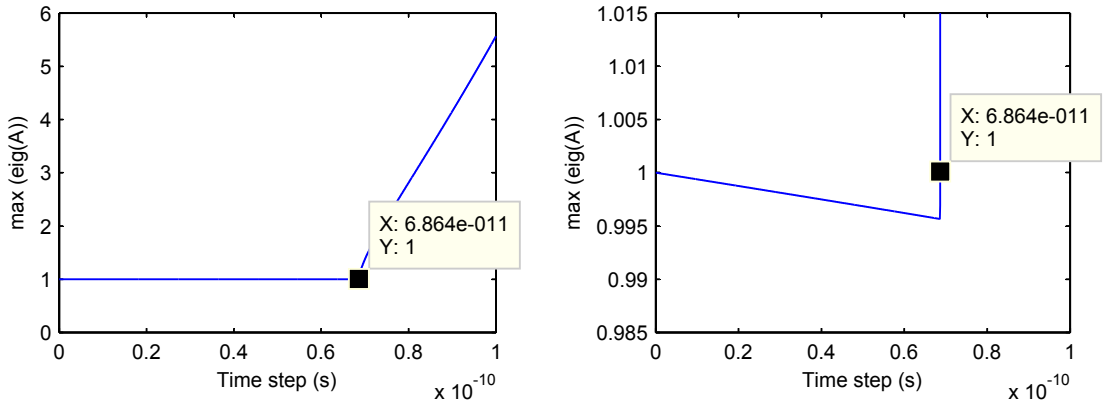


Figure 2.9: Sweep of eigenvalues of the amplification matrix $\mathbf{A}'$. Left: Broad view. Right: Expanded view.

results in a stable and accurate solution, which can be seen from the comparison with SPECTRE shown in Fig. 2.10. On the other hand, the simulation using the time step slightly larger than Δt_{max} results in an unstable simulation as can be seen in Fig. 2.11. Note that selecting a time step to ensure stability does not necessarily

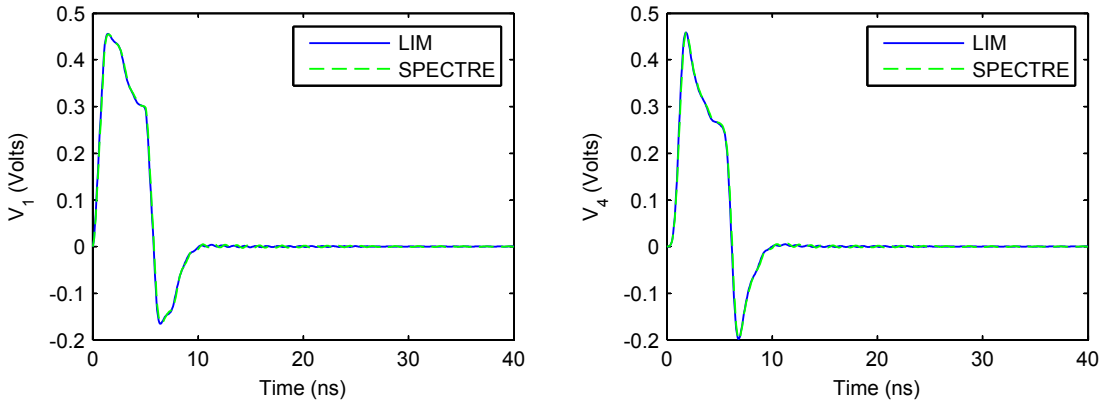


Figure 2.10: Simulation of circuit in Fig. 2.8 with $\Delta t = 6.8 \times 10^{-11}$ s. Left: Voltage at node 1. Right: Voltage at node 4.

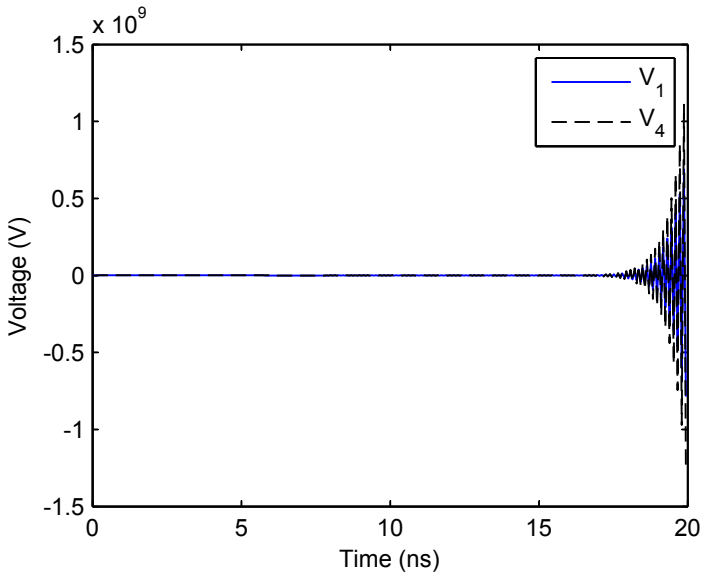


Figure 2.11: Simulation of circuit in Fig. 2.8 with $\Delta t = 6.9 \times 10^{-11}$ s

ensure accuracy. In general, the time step must also be small enough for sufficient accuracy. However, typically for a LIM simulation, the time step to ensure stability is small enough such that the accuracy is also preserved.

CHAPTER 3

PARTITIONED LATENCY INSERTION METHOD (PLIM)

3.1 Introduction

As we have seen in the previous chapter, the LIM algorithm is only conditionally stable, with an upper bound on the maximum time step which depends mainly on the smallest inductance and capacitance in the circuit. When the circuit contains very small latency elements, the time step required for a stable simulation could be equally small, which would result in a large number of time steps in a transient simulation. In order to alleviate this problem, a block processing technique has been proposed [32–34] which utilizes different time steps for different parts of the circuit. However, selecting the maximum time step for each part of the circuit is still a challenging task, and the basic method used in [32–34] to select the time step can only be applied under very restrictive assumptions [35]. Specifically, each node in the circuit has to be connected to only two branches, and the values of the circuit elements have to be the same everywhere in each sub-circuit. In this dissertation, we propose a more robust method to select the maximum time step of the LIM simulation, which is independent of the circuit topology. We apply the amplification matrix, developed in the previous chapter, along with the block processing technique to the simulation of circuits with partitions of different latencies and demonstrate the accuracy and speed improvements of the proposed method over the basic LIM [28, 29].

3.2 Motivation and Method

We first illustrate the motivation of the method via an example. Consider a case where LIM is used to simulate a circuit consisting of a transmission line TLINE 1 connected to a purely resistive external network as shown in Fig. 3.1. The transmission line has RLGC values shown in Fig. 3.1 where for simplicity we have assumed that G = 0 such that the method presented in [36] for selecting the time step for an RLC circuit can be applied. In order to simulate this circuit in LIM, we model the transmission line with 10 segments of RLC lumped elements, and insert fictitious latency elements into the external network as shown in Fig. 3.2, where the fictitious elements have been made small so as to not affect the accuracy of the solution.

The time step required for a stable simulation of this RLC circuit can then be shown to be [36]

$$\begin{aligned} \Delta t &< \sqrt{2} \min_{i=1}^{N_n} \left(\sqrt{\frac{C_i}{M_i} \min_{p=1}^{M_i} (L_{i,p})} \right) \\ &< \sqrt{2} \left(\sqrt{\frac{0.01p}{3} 0.025n} \right) = 4.08 \times 10^{-13} s \end{aligned} \tag{3.1}$$

where $L_{i,p}$ denotes the value of the pth inductor connected to node i. Notice that in this case, the L and C of the external network completely determine the maximum time step. In other words, the maximum time step to ensure stability is dictated by the section with the smallest latency. However, note that if we had considered a circuit with only the transmission line TLINE 1, the maximum time step would have been

$$\Delta t < \sqrt{2.5n \cdot 1p} = 5 \times 10^{-11} s. \tag{3.2}$$

This suggests that the section with higher latency can be simulated with a larger time step without violating the stability criterion.

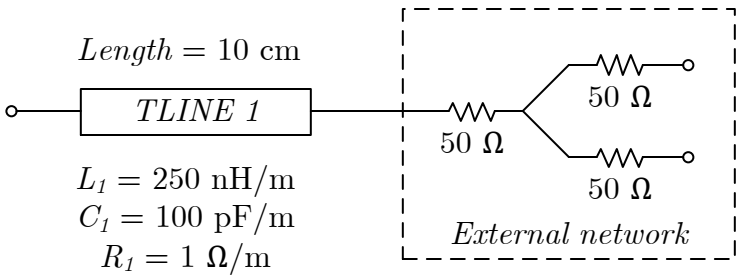


Figure 3.1: Transmission line connected to an external network.

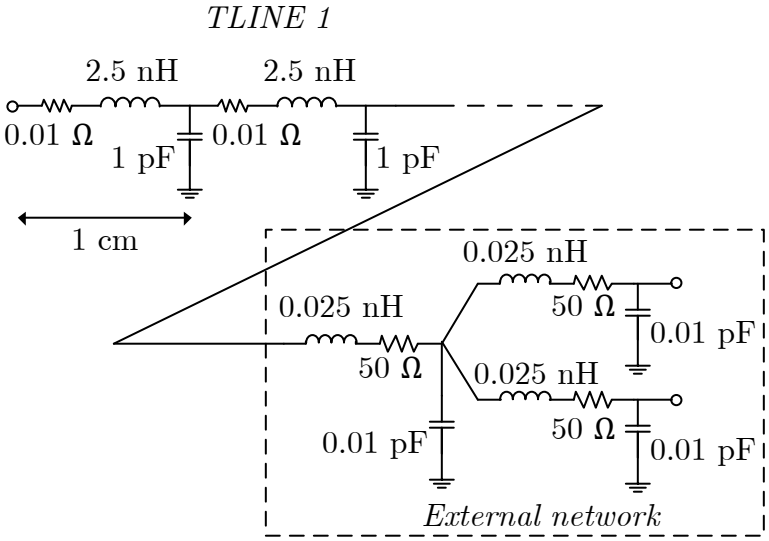


Figure 3.2: LIM enabled circuit of Fig. 3.1.

Consider then the following method for simulating circuits with partitions of different latencies. First, a stable time step is determined for each partition. In the case of an RLC or a GLC circuit, the method in [36, 37] is employed as shown in (3.1). However, for a general circuit (or in the presence of dependent sources), (3.1) cannot be applied and the more general numerical method presented in Theorem 1 must be used. Once all the time steps have been determined, the smallest time step is used in LIM to simulate the circuit, but each partition is only updated as needed, depending on its maximum stable time step. This results in a computationally ef-

ficient algorithm, with large speed-ups in the simulation time, especially when the partition with the smallest latency is small compared to the rest of the circuit. The method is summarized in Fig. 3.3.

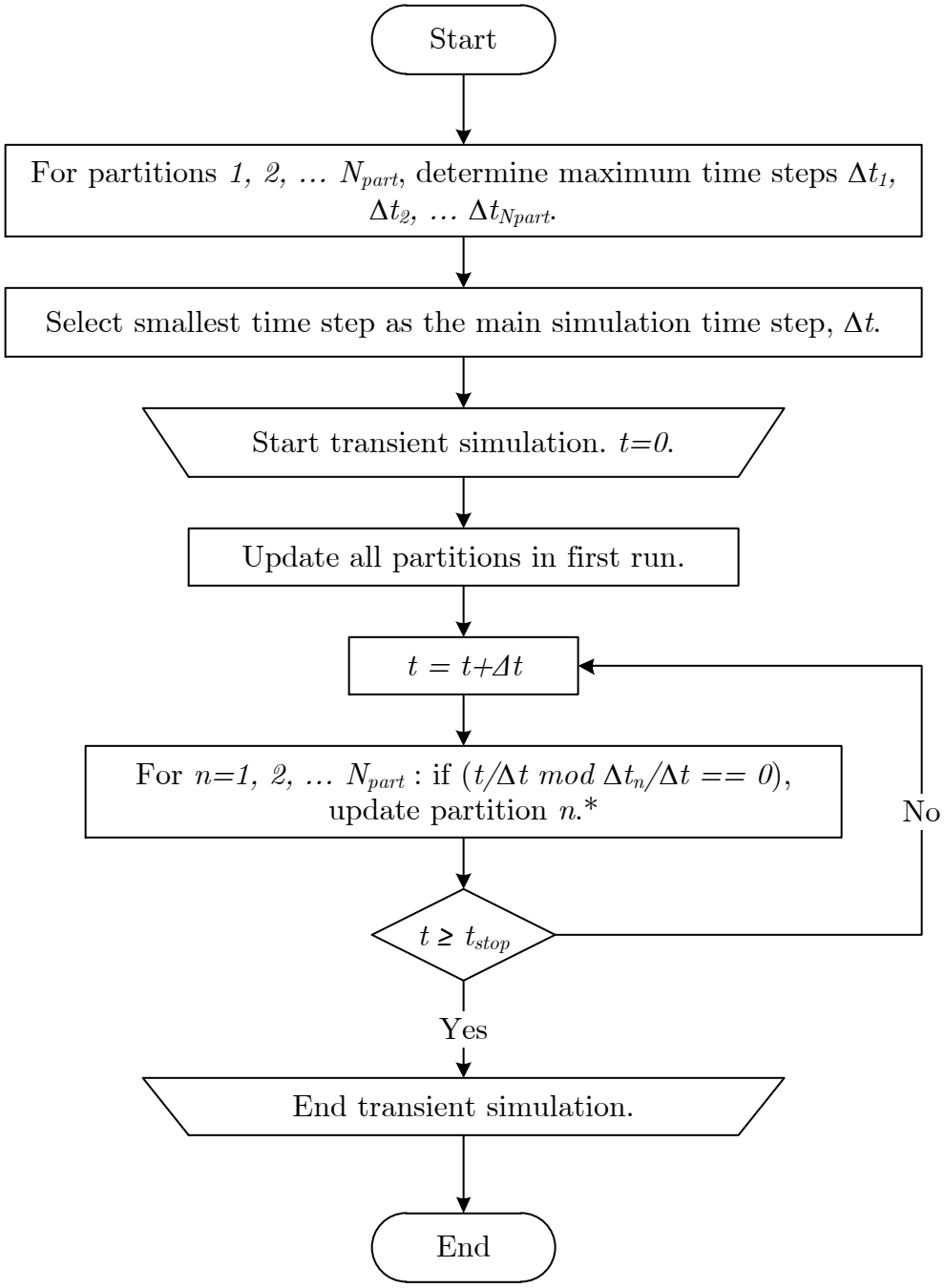
3.3 Example

We present an example to depict the usage of multiple time steps on a circuit with partitions of different latencies. Speed improvements over the conventional LIM will be illustrated. Consider the circuit in Fig. 3.4 which consists of three partitions detailed as follows:

1. Partition 1: High latency partition.
2. Partition 2: Low latency partition. (In practice, this could be a partition with no latency, whereby small fictitious elements have been inserted to enable LIM.)
3. Partition 3: Dependent sources.

The input is a current source with a single trapezoidal pulse of rise and fall times equal to 1 ns and a pulse width of 4 ns. The maximum amplitude is 0.02 A. Using the method in the previous section, the maximum time step of each partition is determined to be $\Delta t_1 = 1.0486 \times 10^{-10}$ s, $\Delta t_2 = 1.07 \times 10^{-12}$ s and $\Delta t_3 = 6.741 \times 10^{-11}$ s corresponding to partitions one, two and three respectively. Note that we have chosen the time steps to be integer multiples of the smallest time step (in this case Δt_2) as mentioned in the previous section.

The circuit is then simulated using the algorithm in Fig. 3.3 for performing LIM with partitions of different latencies (PLIM) and the results at the input (node 1a) and output (node 4c) are shown in Fig. 3.5. Next, the same circuit is simulated using the traditional LIM with time step $\Delta t = \Delta t_2 = 1.07 \times 10^{-12}$ s and the results are also plotted in Fig. 3.5. No loss in accuracy is observed when using PLIM compared to the regular LIM.



* For simplicity, it is assumed that all the time steps are integer multiples of the smallest time step. If not, they are rounded down to the nearest integer multiple of the smallest time step.

Figure 3.3: Simulation algorithm for partitioned LIM.

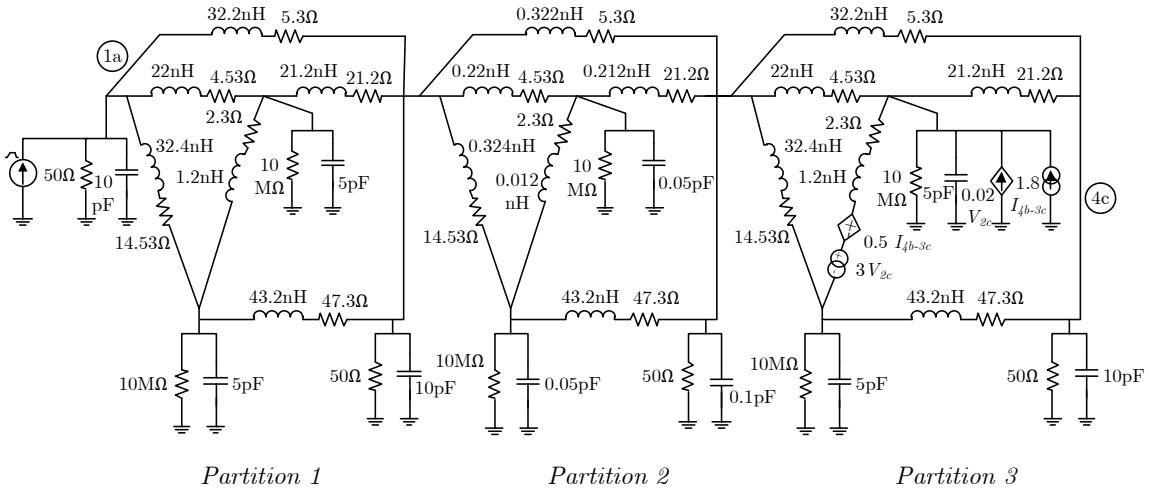


Figure 3.4: Example circuit with partitions of different latencies.

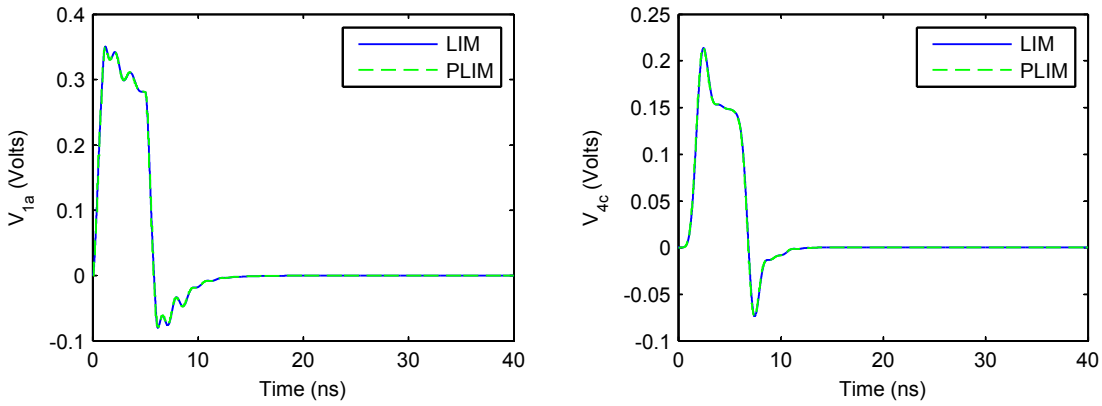


Figure 3.5: Simulation of circuit in Fig. 3.4. Left: Voltage at node 1a. Right: Voltage at node 4c.

Next, PLIM is used to simulate a large circuit where the partition with the smallest latency is small compared to the rest of the circuit. To construct this circuit, partition 1 is cascaded N times and the simulation time is recorded for PLIM and LIM. The results are summarized in Table 3.1. All simulations were performed on a Linux server with Intel Xeon 3.16 GHz processors and 32 GB of RAM. We observe that when the sizes of the partitions are comparable, a small speed-up is obtained when using PLIM. On the other hand, when the partition with the smallest latency is small

Table 3.1: Comparison of runtime for LIM and PLIM.

N	LIM (s)	PLIM (s)	Speed-up
1	0.11	0.01	11.0
10	0.23	0.02	11.5
100	1.40	0.03	46.7
500	7.15	0.08	89.4
1000	13.52	0.15	90.1
2000	26.21	0.28	93.6
5000	64.52	0.68	94.9

compared to the rest of the circuit, a large speed-up is obtained which approaches the limit of $\Delta t_{large}/\Delta t_{min}$, where Δt_{large} is the time step of the largest partition and Δt_{min} is the smallest time step in the circuit, which also dictates the maximum time step of the regular LIM.

CHAPTER 4

BLACKBOX MACROMODELING IN LIM

4.1 Introduction

In the past few years, multiport networks characterized by sampled data or blackbox networks have become more frequent in the analysis and design of high-frequency or high-speed circuits. Blackbox macromodeling techniques improve the efficiency of the simulations by representing complex networks in terms of their terminal transfer functions in the frequency domain, where frequency dependent effects are most effectively characterized; this representation is then converted into a form compatible for incorporation into circuit simulators. Common approaches utilize either model-order reduction (MOR) techniques that use curve fitting methods to approximate the blackbox data in a rational function in terms of poles and residues, which is then converted into a SPICE compatible netlist, or IFFT to convert the frequency-domain data into a time-domain impulse response, which can then be convoluted with the input terminal response. A variation of the MOR technique is to use the poles and residues in a recursive convolution algorithm where the terminal responses are obtained via convolution with decaying exponential functions, which can be done substantially faster than the direct convolution manner.

In this work, we demonstrate the use of two approaches for the modeling of blackbox networks characterized by scattering parameters, and their incorporation into LIM simulations; and we compare the two approaches in terms of computational speed

and efficiency. The two approaches are MOR via vector fitting [8] with passivity enforcement, and a fast convolution approach using S-parameters [38].

4.2 MOR via Vector Fitting

In the MOR method, we seek to express the given frequency response of the black-box, $f(s)$, where $s = j\omega$ and ω is the angular frequency, in the form of a rational function

$$f(s) \approx \sum_{n=1}^{N} \frac{c_n}{s - a_n} + d + sh \tag{4.1}$$

where N is the order, c_n are the residues, a_n are the poles, d is the constant term and h is the linear term. In order to solve for these unknowns, we apply the vector fitting process, which decomposes this nonlinear problem into a set of two linear problems as follows.

4.2.1 Vector Fitting for System Identification

In the first stage, the poles a_n of the system are solved for by introducing an unknown function $\sigma(s)$ where $\sigma(s)$ is defined in its rational form as

$$\sigma(s) = \sum_{n=1}^{N} \frac{\tilde{c}_n}{s - \tilde{a}_n} + 1. \tag{4.2}$$

Notice that the ambiguity in the solution for $\sigma(s)$ is removed by forcing it to approach unity at very high frequencies. Now if we assume that both $\sigma(s)$ and the product of $\sigma(s)$ and $f(s)$ (i.e. $\sigma(s)f(s)$) can be approximated by rational functions using the

same set of poles (in this case $\tilde{a}_n$), we have the augmented problem

$$\begin{bmatrix} \sigma(s)f(s) \\ \sigma(s) \end{bmatrix} \approx \begin{bmatrix} \displaystyle\sum_{n=1}^{N} \frac{c_n}{s-\tilde{a}_n} + d + sh \\ \displaystyle\sum_{n=1}^{N} \frac{\tilde{c}_n}{s-\tilde{a}_n} + 1 \end{bmatrix}. \tag{4.3}$$

Multiplying the second row in (4.3) with $f(s)$ and equating it to the first row gives

$$\left(\sum_{n=1}^{N} \frac{c_n}{s-\tilde{a}_n} + d + sh \right) \approx \left(\sum_{n=1}^{N} \frac{\tilde{c}_n}{s-\tilde{a}_n} + 1 \right) f(s) \tag{4.4}$$

which can be rearranged to read

$$\left(\sum_{n=1}^{N} \frac{c_n}{s-\tilde{a}_n} + d + sh \right) - \left(\sum_{n=1}^{N} \frac{\tilde{c}_n}{s-\tilde{a}_n} \right) f(s) \approx f(s). \tag{4.5}$$

Examining (4.5) reveals that if the poles $\tilde{a}_n$ are fixed beforehand, then (4.5) is linear in terms of the unknowns. Since $f(s)$ is often obtained from a set of tabulated data, and the amount of data points collected normally well exceeds the order of approximation, N, writing (4.5) for each frequency sample point results in an overdetermined set of equations

$$\begin{bmatrix} \frac{1}{s_1-\tilde{a}_1} & \cdots & \frac{1}{s_1-\tilde{a}_N} & s_1 & \frac{-f(s_1)}{s_1-\tilde{a}_1} & \cdots & \frac{-f(s_1)}{s_1-\tilde{a}_N} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{s_k-\tilde{a}_1} & \cdots & \frac{1}{s_k-\tilde{a}_N} & s_k & \frac{-f(s_k)}{s_k-\tilde{a}_1} & \cdots & \frac{-f(s_k)}{s_k-\tilde{a}_N} \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_N \\ d \\ h \\ \tilde{c}_1 \\ \vdots \\ \tilde{c}_N \end{bmatrix} = \begin{bmatrix} f(s_1) \\ \vdots \\ \\ \vdots \\ f(s_k) \end{bmatrix} \tag{4.6}$$

where k is the number of frequency sample points. This overdetermined set of equations is in the form of

$$Ax = b \tag{4.7}$$

which can be solved using any standard least squares method for the unknown solution vector x that contains the residues.

Now returning to (4.4) and solving for $f(s)$ gives

$$f(s) \approx \frac{\sum_{n=1}^{N} \frac{c_n}{s - \tilde{a}_n} + d + sh}{\sum_{n=1}^{N} \frac{\tilde{c}_n}{s - \tilde{a}_n} + 1} \tag{4.8}$$

which can be written as a fraction to obtain

$$f(s) \approx \frac{\frac{\prod_{n=1}^{N+1} (s - z_n)}{\prod_{n=1}^{N} (s - \tilde{a}_n)}}{\frac{\prod_{n=1}^{N} (s - \tilde{z}_n)}{\prod_{n=1}^{N} (s - \tilde{a}_n)}} = \frac{\prod_{n=1}^{N+1} (s - z_n)}{\prod_{n=1}^{N} (s - \tilde{z}_n)} \tag{4.9}$$

where we see that the poles of $f(s)$ become equal to $\tilde{z}_n$ which are the zeros of $\sigma(s)$, and the initial poles $\tilde{a}_n$ are cancelled out in the process.

In order to solve for the zeros of $\sigma(s)$ (which are the poles of $f(s)$), we define a state space system for $\sigma(s)$ in the form of

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{4.10}$$

where x is the state vector with $\dot{x} = dx/dt$, u is the input vector and y is the output vector.

For the function $\sigma(s)$, A is a diagonal matrix holding the poles $\tilde{a}_n$, B is a column vector of ones, C is a row vector holding its residues $\tilde{c}_n$, and D is unity. In order to solve for the zeros of $\sigma(s)$, we return to (4.2) and rewrite it in the form of a fraction:

$$\sigma(s) = \sum_{n=1}^{N} \frac{\tilde{c}_n}{s - \tilde{a}_n} + 1 = \frac{\prod_{n=1}^{N} (s - \tilde{z}_n)}{\prod_{n=1}^{N} (s - \tilde{a}_n)} = \frac{y(s)}{u(s)}. \tag{4.11}$$

Notice that the zeros of $\sigma(s)$ are equal to the poles of $1/\sigma(s)$. With (4.11) as the representation for $\sigma(s)$, we can obtain the expression for $1/\sigma(s)$ by interchanging the input and the output. Solving for u in the second equation in (4.11) and plugging it into the first, we obtain

$$u = D^{-1}(y - Cx) \tag{4.12}$$

$$\begin{aligned} \dot{x} &= Ax + BD^{-1}(y - Cx) = Ax + BD^{-1}y - BD^{-1}Cx \\ &= (A - BD^{-1}C)x + BD^{-1}y. \end{aligned} \tag{4.13}$$

Thus the poles of $1/\sigma(s)$ can be calculated as

$$eig(A - BD^{-1}C) \tag{4.14}$$

which simplifies to

$$eig(A - BC) \tag{4.15}$$

since D is unity.

In summary, the poles of $f(s)$ can be calculated as the eigenvalues of the matrix $(A - BC)$ where

$$A = \begin{bmatrix} \tilde{a}_1 & & 0 \\ & \ddots & \\ 0 & & \tilde{a}_N \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \qquad C = \begin{bmatrix} \tilde{c}_1 & \cdots & \tilde{c}_N \end{bmatrix}. \tag{4.16}$$

Once the poles of the system are solved for, the residues can be easily solved by returning to (4.1) which is now linear in terms of the unknowns c_n, d and h. We proceed as before by writing (4.1) for each frequency sample point to obtain:

$$\begin{bmatrix} \frac{1}{s_1 - a_1} & \cdots & \frac{1}{s_1 - a_N} & 1 & s_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{1}{s_k - a_1} & \cdots & \frac{1}{s_k - a_N} & 1 & s_k \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_N \\ d \\ h \end{bmatrix} = \begin{bmatrix} f(s_1) \\ \vdots \\ \\ \vdots \\ f(s_k) \end{bmatrix}. \tag{4.17}$$

This again results in an overdetermined set of equations which can be solved as before for the residues c_n, constant d, and linear term h.

This completes the vector fitting process to find a rational function approximation for the blackbox in the form of (4.1). We see that the solution is not guaranteed to be exact but instead depends on minimizing the error of a set of two least squares problems. Thus at this point, one would normally compare the approximation to the original data and determine if they are within acceptable range. If necessary, a more accurate solution can be obtained if the vector fitting algorithm is repeated on the data by using the *newly calculated poles as starting poles*. Therefore, vector fitting is often seen as an iterative scheme whereby the poles are relocated until they converge

with the actual poles of the system. Normally this is achieved rather quickly and it takes on average 2 – 4 iterations to obtain an accurate result.

4.2.1.1 Modification for Complex Poles

An important modification to the vector fitting algorithm which is often made when solving for real systems with complex poles will now be presented. For real systems, the poles must either be real or in complex-conjugate pairs. In addition, the residues corresponding to the real poles must be real and, similarly, the residues corresponding to the complex-conjugate pair poles must also be in complex-conjugate pairs. In order to make the necessary adjustment to (4.6) to ensure this condition, we return to (4.5) and rewrite it for systems with both real and complex poles.

Assume a system with Q real poles and L complex-conjugate pole pairs where an asterisk "*" is used as a notation to indicate complex-conjugacy. Thus for a complex pair, we would have

$$a_n = a_n^r + ja_n^i \ , \quad a_n^* = a_n^r - ja_n^i \tag{4.18}$$

$$c_n = c_n^r + jc_n^i \ , \quad c_n^* = c_n^r - jc_n^i \tag{4.19}$$

with the superscript r representing the real part and the superscript i representing the imaginary part. Equation (4.5) now becomes

$$\begin{aligned} &\left[\sum_{q=1}^{Q} \frac{c_q}{s-\tilde{a}_q} + \sum_{l=1}^{L}\left(\frac{c_l}{s-\tilde{a}_l} + \frac{c_l^*}{s-\tilde{a}_l^*}\right) + d + sh\right] \\ &- \left[\sum_{q=1}^{Q} \frac{\tilde{c}_q}{s-\tilde{a}_q} + \sum_{l=1}^{L}\left(\frac{\tilde{c}_l}{s-\tilde{a}_l} + \frac{\tilde{c}_l^*}{s-\tilde{a}_l^*}\right)\right] \cdot f(s) \approx f(s). \end{aligned} \tag{4.20}$$

Since each complex pair consists of two poles, we have that the order of approximation $N = Q + 2L$.

The elements of the matrix A in (4.7) then become

$$A_{k,q} = \frac{1}{s_k - \tilde{a}_q} \tag{4.21}$$

for each of the real poles and

$$A_{k,l} = \frac{1}{s_k - \tilde{a}_l} + \frac{1}{s_k - \tilde{a}_l^*}\ , \quad A_{k,l+1} = \frac{j}{s_k - \tilde{a}_l} - \frac{j}{s_k - \tilde{a}_l^*} \tag{4.22}$$

for each of the complex pole pairs.

Writing (4.20) for each frequency sample point with the help of (4.21) and (4.22), gives

$$\begin{bmatrix} & & 1 & s_1 & & \\ [R] & [C] & \vdots & \vdots & [G] & [H] \\ & & 1 & s_k & & \end{bmatrix} [x] = \begin{bmatrix} f(s_1) \\ \vdots \\ f(s_k) \end{bmatrix} \tag{4.23}$$

where the matrices $[R]$, $[C]$, $[G]$ and $[H]$ are as follows:

$$R = \begin{bmatrix} \frac{1}{s_1-\tilde{a}_1} & \cdots & \frac{1}{s_1-\tilde{a}_Q} \\ \vdots & \ddots & \vdots \\ \frac{1}{s_k-\tilde{a}_1} & \cdots & \frac{1}{s_k-\tilde{a}_Q} \end{bmatrix} \tag{4.24}$$

$$C = \begin{bmatrix} \frac{1}{s_1-\tilde{a}_1} + \frac{1}{s_1-\tilde{a}_1^*} & \frac{j}{s_1-\tilde{a}_1} - \frac{j}{s_1-\tilde{a}_1^*} & \cdots & \frac{1}{s_1-\tilde{a}_L} + \frac{1}{s_1-\tilde{a}_L^*} & \frac{j}{s_1-\tilde{a}_L} - \frac{j}{s_1-\tilde{a}_L^*} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{s_k-\tilde{a}_1} + \frac{1}{s_k-\tilde{a}_1^*} & \frac{j}{s_k-\tilde{a}_1} - \frac{j}{s_k-\tilde{a}_1^*} & \cdots & \frac{1}{s_k-\tilde{a}_L} + \frac{1}{s_k-\tilde{a}_L^*} & \frac{j}{s_k-\tilde{a}_L} - \frac{j}{s_k-\tilde{a}_L^*} \end{bmatrix} \tag{4.25}$$

$$G = \begin{bmatrix} \frac{-f(s_1)}{s_1-\tilde{a}_1} & \cdots & \frac{-f(s_1)}{s_1-\tilde{a}_Q} \\ \vdots & \ddots & \vdots \\ \frac{-f(s_k)}{s_k-\tilde{a}_1} & \cdots & \frac{-f(s_k)}{s_k-\tilde{a}_Q} \end{bmatrix} \tag{4.26}$$

$$H = \begin{bmatrix} \frac{-f(s_1)}{s_1-\tilde{a}_1} + \frac{-f(s_1)}{s_1-\tilde{a}_1^*} & \frac{-jf(s_1)}{s_1-\tilde{a}_1} - \frac{-jf(s_1)}{s_1-\tilde{a}_1^*} & \cdots & \frac{-f(s_1)}{s_1-\tilde{a}_L} + \frac{-f(s_1)}{s_1-\tilde{a}_L^*} & \frac{-jf(s_1)}{s_1-\tilde{a}_L} - \frac{-jf(s_1)}{s_1-\tilde{a}_L^*} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{-f(s_k)}{s_k-\tilde{a}_1} + \frac{-f(s_k)}{s_k-\tilde{a}_1^*} & \frac{-jf(s_k)}{s_k-\tilde{a}_1} - \frac{-jf(s_k)}{s_k-\tilde{a}_1^*} & \cdots & \frac{-f(s_k)}{s_k-\tilde{a}_L} + \frac{-f(s_k)}{s_k-\tilde{a}_L^*} & \frac{-jf(s_k)}{s_k-\tilde{a}_L} - \frac{-jf(s_k)}{s_k-\tilde{a}_L^*} \end{bmatrix}. \tag{4.27}$$

Notice that when there are no complex poles (i.e., $L = 0$), the matrices $[C]$ and $[H]$ become the empty matrix and (4.23) reduces to (4.6) with $N = Q$.

Finally, we formulate (4.23) in terms of real quantities as

$$\begin{bmatrix} \mathrm{Re}(A) \\ \mathrm{Im}(A) \end{bmatrix} \cdot [x] = \begin{bmatrix} \mathrm{Re}(b) \\ \mathrm{Im}(b) \end{bmatrix} \tag{4.28}$$

and solve for the solution vector x to yield

$$x = \begin{bmatrix} c_1 & \cdots & c_Q & c_1^r & c_1^i & \cdots & c_L^r & c_L^i & d & h & \tilde{c}_1 & \cdots & \tilde{c}_Q & \tilde{c}_1^r & \tilde{c}_1^i & \cdots & \tilde{c}_L^r & \tilde{c}_L^i \end{bmatrix}^T \tag{4.29}$$

where all the elements are purely real. The complex residues are then formed from (4.19), where we would have

$$c_l = c_l^r + jc_l^i\,, \quad c_{l+1} = c_l^* = c_l^r - jc_l^i \tag{4.30}$$

$$\tilde{c}_l = \tilde{c}_l^r + j\tilde{c}_l^i\,, \quad \tilde{c}_{l+1} = \tilde{c}_l^* = \tilde{c}_l^r - j\tilde{c}_l^i. \tag{4.31}$$

The poles of the system can then be solved as before from (4.15). For each of the complex poles, we modify the corresponding submatrices via a similarity transformation to obtain

$$\hat{A} = \begin{bmatrix} \mathrm{Re}(\tilde{a}) & \mathrm{Im}(\tilde{a}) \\ -\mathrm{Im}(\tilde{a}) & \mathrm{Re}(\tilde{a}) \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad \hat{C} = \begin{bmatrix} \mathrm{Re}(\tilde{c}) & \mathrm{Im}(\tilde{c}) \end{bmatrix}. \tag{4.32}$$

As a result, the matrices now have real coefficients and any complex eigenvalue will come along with its complex-conjugate pair, thus preserving the properties of a real system.

Once the poles of the system are solved for, the residues can then be calculated as before. With the modification for complex poles, (4.17) now becomes

$$\begin{bmatrix} & & 1 & s_1 \\ [R] & [C] & \vdots & \vdots \\ & & 1 & s_k \end{bmatrix} [x] = \begin{bmatrix} f(s_1) \\ \vdots \\ f(s_k) \end{bmatrix} \tag{4.33}$$

where the matrices $[R]$ and $[C]$ are the same as in (4.24) and (4.25), respectively. Solving (4.33) yields the unknown vector x in the form of

$$x = \begin{bmatrix} c_1 & \cdots & c_Q & c_1^r & c_1^i & \cdots & c_L^r & c_L^i & d & h \end{bmatrix}^T \tag{4.34}$$

and the complex residues can be formed as

$$c_l = c_l^r + jc_l^i\,, \quad c_{l+1} = c_l^* = c_l^r - jc_l^i. \tag{4.35}$$

4.2.1.2 Modification for Fitting Vector Functions

So far we have considered the case for fitting a scalar or a single function. However, it is sometimes desirable to fit a vector or multiple functions using the same set of poles since this would result in an increase in efficiency in the time-domain convolutions. The modification for fitting vectors is rather straightforward and is presented below.

Consider a vector of Nc functions:

$$f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{Nc} \end{bmatrix}. \tag{4.36}$$

For this function, (4.5) now becomes

$$\begin{bmatrix} \sum_{n=1}^{N} \frac{c_n^1}{s-\tilde{a}_n} + d^1 + sh^1 \\ \sum_{n=1}^{N} \frac{c_n^2}{s-\tilde{a}_n} + d^2 + sh^2 \\ \vdots \\ \sum_{n=1}^{N} \frac{c_n^{Nc}}{s-\tilde{a}_n} + d^{Nc} + sh^{Nc} \end{bmatrix} - \begin{bmatrix} f_1 \sum_{n=1}^{N} \frac{\tilde{c}_n}{s-\tilde{a}_n} \\ f_2 \sum_{n=1}^{N} \frac{\tilde{c}_n}{s-\tilde{a}_n} \\ \vdots \\ f_{Nc} \sum_{n=1}^{N} \frac{\tilde{c}_n}{s-\tilde{a}_n} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{Nc} \end{bmatrix} \tag{4.37}$$

and (4.6) becomes

$$\begin{bmatrix} [X_{\sigma f}] & 0 & 0 & 0 & -f_1 [X_\sigma] \\ 0 & [X_{\sigma f}] & 0 & 0 & -f_2 [X_\sigma] \\ 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & [X_{\sigma f}] & -f_{Nc} [X_\sigma] \end{bmatrix} \begin{bmatrix} [Y_1] \\ [Y_2] \\ \vdots \\ [Y_{Nc}] \\ \left[\tilde{Y}\right] \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{Nc} \end{bmatrix} \tag{4.38}$$

where

$$X_{\sigma f} = \begin{bmatrix} \frac{1}{s_1 - \tilde{a}_1} & \cdots & \frac{1}{s_1 - \tilde{a}_N} & 1 & s_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{1}{s_k - \tilde{a}_1} & \cdots & \frac{1}{s_k - \tilde{a}_N} & 1 & s_k \end{bmatrix} \tag{4.39}$$

$$X_{\sigma} = \begin{bmatrix} \frac{1}{s_1 - \tilde{a}_1} & \cdots & \frac{1}{s_1 - \tilde{a}_N} \\ \vdots & \ddots & \vdots \\ \frac{1}{s_k - \tilde{a}_1} & \cdots & \frac{1}{s_k - \tilde{a}_N} \end{bmatrix} \tag{4.40}$$

$$Y_{nc} = \begin{bmatrix} c_1^{nc} & \cdots & c_N^{nc} & d^{nc} & h^{nc} \end{bmatrix}^T, \qquad nc \in 1, 2 \ldots Nc \tag{4.41}$$

$$\tilde{Y} = \begin{bmatrix} \tilde{c}_1 & \cdots & \tilde{c}_N \end{bmatrix}^T. \tag{4.42}$$

We can then solve (4.38) for $\tilde{Y}$ and solve for the poles using (4.15) where the matrix elements are given in (4.16). This has the effect that a single set of poles that minimizes the least squares error in all elements of (4.36) is obtained. The residues of the individual functions can then be solved for by carrying out (4.17) independently for each element in (4.36). It should also be noted that if complex poles are used, the modifications presented in the previous section should also be carried out for each element of the vector.

4.2.1.3 Modification for Fast Fitting Vector Functions

In this section, we review a recent modification to the vector fitting process for the fast fitting of vector functions [11]. As presented in the previous section, the first step of the vector fitting method is to solve for the residues of $\sigma(s)$ from an overdetermined set of equations. When fitting multiple functions using the same set of poles, the size of this overdetermined set of equations may get prohibitively large. However, note that only part of the solution vector was needed. For example, in (4.38), only the

solution vector $\tilde{Y}$ was needed while the others $(Y_1 - Y_{Nc})$ were discarded. A more efficient formulation is possibly by first applying a QR decomposition to the least squares equations of each of the elements

$$[[X_{\sigma f}] - f\,[X_\sigma]] = [Q] \begin{bmatrix} R^{11} & R^{12} \\ R^{21} & R^{22} \end{bmatrix}. \tag{4.43}$$

Once all the Q and R submatrices have been extracted, an overall overdetermined set of equations is formed to solve for the residues of $\sigma(s)$ as

$$\begin{bmatrix} R_1^{22} \\ R_2^{22} \\ \vdots \\ R_{Nc}^{22} \end{bmatrix} \left[\tilde{Y}\right] = \begin{bmatrix} Q_1^T f_1 \\ Q_2^T f_2 \\ \vdots \\ Q_{Nc}^T f_{Nc} \end{bmatrix}. \tag{4.44}$$

This has the effect that the new overdetermined set of equations is now significantly smaller than before and the solution vector is only the residues of $\sigma(s)$. Although it requires solving the QR decomposition of each individual element to be fitted, that process is often less time-consuming since the matrices are much smaller. When the number of elements to be fitted increases, for example in multiport devices with a large number of ports, the computational savings could be enormous.

4.2.1.4 Stability and Starting Poles Selections

In this section, we present a brief discussion regarding stability and starting pole selection. For a causal system to be stable, its poles must lie in the left half-plane of the s-domain. In the vector fitting process, however, it is possible to obtain unstable poles when solving (4.15). This can easily be corrected either by discarding any

unstable poles that were obtained from (4.15) or by flipping unstable poles into the left half-plane by reflecting it on the imaginary axis of the s-domain.

The vector fitting method requires that an initial set of starting poles be specified for use as a preliminary guess of the actual poles. Although these starting poles cancel out in the subsequent formulation, a poor choice of these values can result in a large variation between the original function and the fitted function as the vector fitting method relies on solving (4.7) in a least squares sense. Starting pole selection methods are heuristic at best, and a good rule of thumb is to select the starting poles to be in complex conjugate pairs situated along a line close to the imaginary axis [8].

4.2.1.5 Summary

An overall flowchart showing the whole vector fitting process is shown in Fig. 4.1.

4.2.2 Passivity Enforcement

Passivity is defined as the inability of the system to generate energy in any termination condition [39]. If the system being modeled is passive, then the macromodel generated must be passive as well, since stable but nonpassive models can result in unstable systems when connected to other passive components [40]. Thus, ensuring passivity of the model is a crucial step in the macromodel generation process.

A precise mathematical definition of passivity depends on the adopted representation. For a system characterized by the scattering parameters $S(s)$, the condition for passivity is [14]

1. $S(s^*) = S^*(s)$ where "*" denotes the complex conjugate operator.
2. $S(s)$ is bounded real.

   I.e., $\|S(j\omega)\| \leqslant 1$ or $eig\left(I - S(j\omega)^H S(j\omega)\right) \geqslant 0, \qquad \omega \in \mathbb{R}$

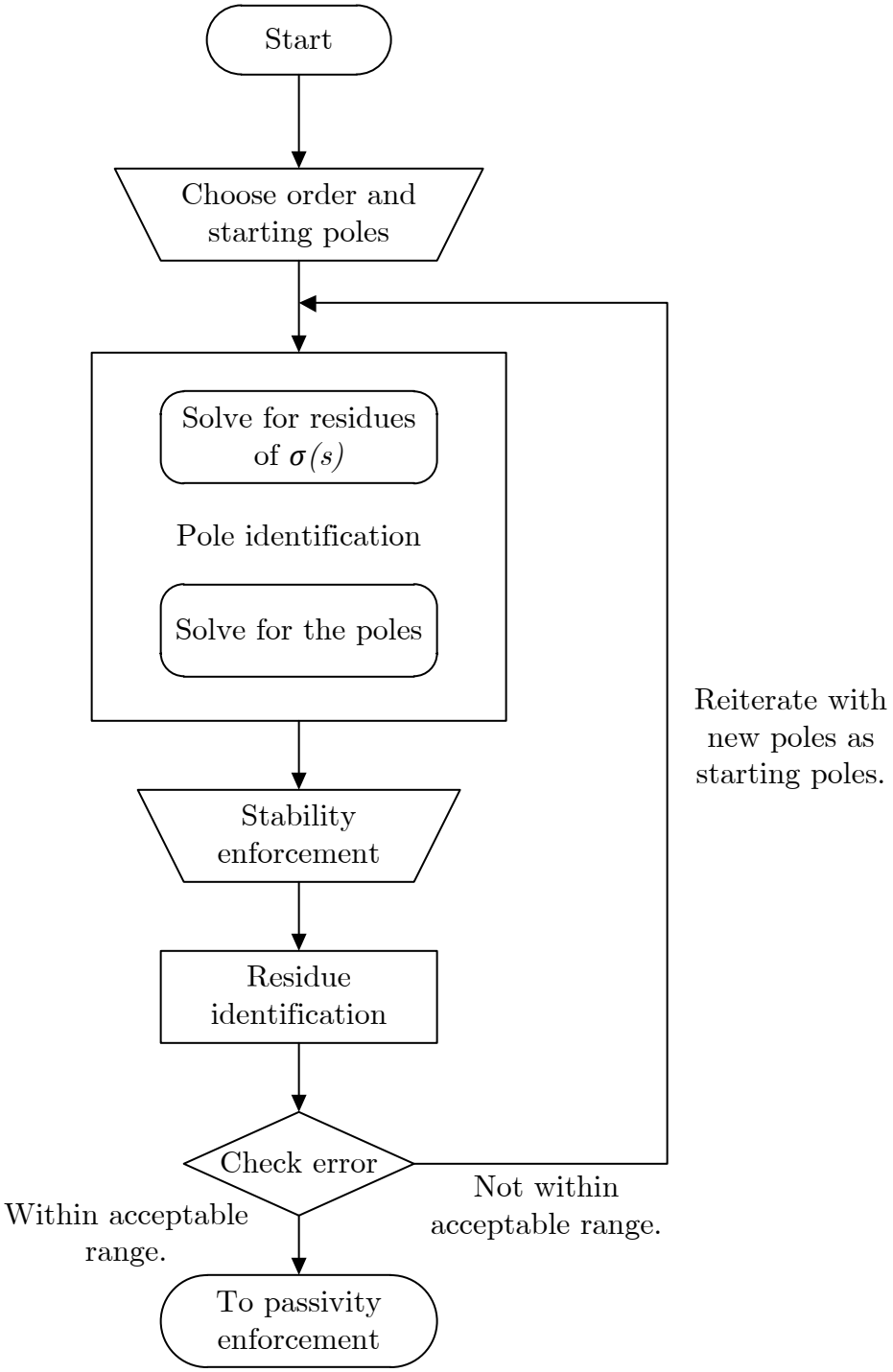


Figure 4.1: Flowchart of the vector fitting process.

Condition 1 is always satisfied in our macromodel since, in the vector fitting process, the complex poles and residues are always considered along with their conjugates, thus leading to only real coefficients in $S(s)$. Consequently, enforcing passivity of the macromodel then amounts to enforcing condition 2.

4.2.2.1 Passivity Assessment

Different methods exist to assess passivity. Recently the method based on the Hamiltonian matrix has gained popularity as a passivity assessment tool since it is independent of frequency and provides the exact bands of violations when passivity violations are detected. For a system described in state-space form as given in (4.10), the system is passive if and only if the Hamiltonian matrix $\mathbf{M}$ has no imaginary eigenvalues [41], where the Hamiltonian is given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} + \mathbf{B}\mathbf{K}\mathbf{D}^T\mathbf{C} & \mathbf{B}\mathbf{K}\mathbf{B}^T \\ -\mathbf{C}^T\mathbf{L}\mathbf{C} & -\mathbf{A}^T - \mathbf{C}^T\mathbf{D}\mathbf{K}\mathbf{B}^T \end{bmatrix} \tag{4.45}$$

where

$$\begin{aligned} \mathbf{K} &= \left(\mathbf{I} - \mathbf{D}^T\mathbf{D}\right)^{-1} \\ \mathbf{L} &= \left(\mathbf{I} - \mathbf{D}\mathbf{D}^T\right)^{-1} \end{aligned} \tag{4.46}$$

where the superscript T is used for the transpose and $\mathbf{I}$ is the identity matrix of appropriate dimensions. In addition, imaginary eigenvalues correspond to points where eigenvalues of the dissipation matrix $I - S(j\omega)^H S(j\omega)$ are equal to zero, therefore defining potential crossover frequencies at which the system switches from being passive to nonpassive (or vice versa). Thus they can be used to define the bands of passivity violations, as will be illustrated next.

Consider a plot of the eigenvalues of the dissipation matrix $I - S(j\omega)^H S(j\omega)$ of a general m-port scattering matrix shown in Fig. 4.2 (note that only plots of two

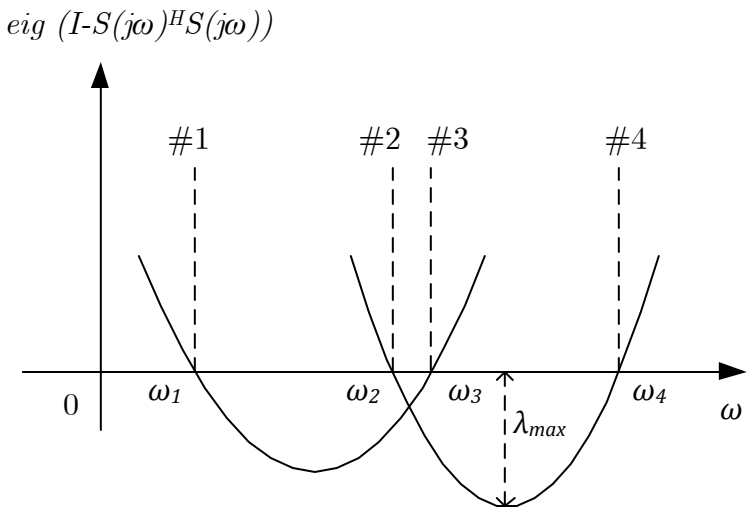


Figure 4.2: Determination of the band of passivity violation.

eigenvalues are shown). From Fig. 4.2, we see that there are four points (marked #1 to #4) where the eigenvalues of the dissipation matrix are equal to zero, thus defining potential points where the system crosses from being passive to nonpassive. As mentioned before, these points are obtained from the eigenvalues of the Hamiltonian matrix that are purely imaginary. In order to obtain the bands of passivity violations, we return to condition 2 above and check whether or not the system is passive at a short distance right before and after the potential crossover frequency. If the system is found to be passive right before the point of consideration but not passive after, the point is defined as a crossover frequency where the system crosses from being passive to nonpassive (i.e., point #1 in Fig. 4.2). If the system is not passive right before the point of consideration but is passive after, the point is defined as a crossover frequency where the system crosses from being nonpassive to passive (i.e., point #4 in Fig. 4.2). On the other hand, if the system is both nonpassive right before and after the point of consideration, then it is concluded that the point is contained within a larger passivity violation band due to the other eigenvalues (i.e., points #2 and #3 in Fig. 4.2). Thus we are able to determine the exact band of passivity violation by

arranging the points in order and determining all the crossover frequencies. In the example given in Fig. 4.2, the band would be from ω_1 to ω_4.

So far we have seen how the bands of passivity violations are determined with the use of the Hamiltonian matrix. Before we proceed to the passivity enforcement section, let us see how two other important quantities which are needed for passivity enforcement are determined. These two quantities are the frequency of maximum violation and the magnitude of maximum violation in each violation band. These locations can be found by solving

$$\lambda = \max \left| eig \left(I - S(j\omega)^H S(j\omega) \right) \right|, \qquad \omega \in \omega_l, \omega_h \tag{4.47}$$

where λ is the magnitude of maximum violation and ω_l and ω_h are the boundaries of the passivity violation band. This is easily solved by doing a fine sweep of each frequency violation band that was found and recording the maximum value and the corresponding frequency point as given in (4.47). With this information, we are now able to proceed to enforce passivity for nonpassive systems.

4.2.2.2 Passivity Enforcement

Passivity enforcement can be performed by a number of methods. For example, in [42] passivity enforcement is performed by perturbing the residues of the system while in [13] passivity is restored by perturbing the Hamiltonian matrix. We adopt a residue perturbation scheme similar to that in [42]. This is presented next.

Consider again a system in state space form as given in (4.10). The scattering matrix of this system can be obtained from

$$\mathbf{S}(j\omega) = \mathbf{C}(j\omega\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}. \tag{4.48}$$

For this system to be passive, it must obey condition 2 presented in the previous section

$$eig(\mathbf{Q}(j\omega)) \geq 0 \tag{4.49}$$

at all frequency points, where $\mathbf{Q}(j\omega)$ denotes the dissipation matrix given by

$$\mathbf{Q}(j\omega) = \mathbf{I} - \mathbf{S}^H(j\omega)\mathbf{S}(j\omega) \tag{4.50}$$

where the superscript H is used to denote the complex conjugate transpose.

If the system is nonpassive, we can attempt to restore passivity by perturbing the representation of the system given in (4.48) by a small amount such that the new system satisfies (4.49) at the frequency points of violation.

Consider a small perturbation $\mathbf{\Delta C}$ to the residue matrix $\mathbf{C}$ which results in a change in the scattering matrix:

$$\hat{\mathbf{S}}(j\omega) = (\mathbf{C} + \mathbf{\Delta C})\,(j\omega\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} \tag{4.51}$$

which can be written as

$$\hat{\mathbf{S}}(j\omega) = \mathbf{S}(j\omega) + \mathbf{\Delta S}(j\omega) \tag{4.52}$$

where

$$\mathbf{\Delta S}(j\omega) = \mathbf{\Delta C}(j\omega\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \mathbf{\Delta C}\mathbf{V} \tag{4.53}$$

with

$$\mathbf{V} = (j\omega\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}. \tag{4.54}$$

In order to ensure passivity, this perturbed scattering matrix must obey

$$eig\left(\hat{\mathbf{Q}}(j\omega)\right) \geq 0 \tag{4.55}$$

where $\hat{\mathbf{Q}}(j\omega)$ is the perturbed dissipation matrix

$$\hat{\mathbf{Q}}(j\omega) = \mathbf{I} - \hat{\mathbf{S}}^H(j\omega)\hat{\mathbf{S}}(j\omega). \tag{4.56}$$

Substituting (4.52) into (4.56) gives (dropping $j\omega$ for simplicity)

$$\hat{\mathbf{Q}} = \mathbf{I} - \hat{\mathbf{S}}^H\hat{\mathbf{S}} = \mathbf{I} - \mathbf{S}^H\mathbf{S} - \mathbf{S}^H\boldsymbol{\Delta}\mathbf{S} - \boldsymbol{\Delta}\mathbf{S}^H\mathbf{S} - \boldsymbol{\Delta}\mathbf{S}^H\boldsymbol{\Delta}\mathbf{S}. \tag{4.57}$$

Neglecting the second-order term in (4.57), we get

$$\hat{\mathbf{Q}} \simeq \mathbf{I} - \mathbf{S}^H\mathbf{S} - \mathbf{S}^H\boldsymbol{\Delta}\mathbf{S} - \boldsymbol{\Delta}\mathbf{S}^H\mathbf{S}. \tag{4.58}$$

Comparing (4.58) to (4.50) reveals that the perturbation results in a change of

$$\boldsymbol{\Delta}\mathbf{Q} = -\mathbf{S}^H\boldsymbol{\Delta}\mathbf{S} - \boldsymbol{\Delta}\mathbf{S}^H\mathbf{S} \tag{4.59}$$

from the unperturbed system. Thus, if the unperturbed nonpassive system violates (4.49) at a particular frequency by an amount λ, we can restore passivity at that point by perturbing the system such that the change in the dissipation matrix given by (4.59) results in a change of its eigenvalue by an amount equal and opposite to λ. To do this, we invoke the first-order eigenvalue perturbation formula [43] which states that a matrix $\mathbf{K}$ perturbed by an amount $\boldsymbol{\Delta}\mathbf{K}$ will result in a change of $\Delta\lambda$ in its eigenvalue given by

$$\Delta\lambda = \frac{\mathbf{y}^T\boldsymbol{\Delta}\mathbf{K}\mathbf{x}}{\mathbf{y}^T\mathbf{x}} \tag{4.60}$$

where $\mathbf{y}$ and $\mathbf{x}$ are the left and right eigenvectors of $\mathbf{K}$, respectively. Therefore, a matrix $\mathbf{Q}$ given in (4.50) perturbed by an amount $\boldsymbol{\Delta}\mathbf{Q}$ given by (4.59) would result

in a change in its eigenvalue by an amount

$$\Delta\lambda = \frac{\mathbf{v}^T \left(-\mathbf{S}^H \mathbf{\Delta S} - \mathbf{\Delta S}^H \mathbf{S}\right) \mathbf{u}}{\mathbf{v}^T \mathbf{u}} \tag{4.61}$$

where $\mathbf{v}$ and $\mathbf{u}$ are the left and right eigenvectors of $\mathbf{Q}$, respectively. Since for a matrix $\mathbf{A}$, the eigenvalues and eigenvectors can be solved such that $\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}$ where $\mathbf{V}$ is a modal matrix (its columns are the eigenvectors of $\mathbf{A}$) and $\mathbf{D}$ is the canonical form of $\mathbf{A}$ (a diagonal matrix with the eigenvalues of $\mathbf{A}$ on the main diagonal), the eigenvectors can be scaled such that $\mathbf{v}^T\mathbf{u}$ would result in unity for a given eigenvalue. Thus, dropping the term $\mathbf{v}^T\mathbf{u}$ and substituting (4.53) in (4.61) gives

$$\Delta\lambda = \mathbf{v}^T \left(-\mathbf{S}^H \mathbf{\Delta C V} - (\mathbf{\Delta C V})^H \mathbf{S}\right) \mathbf{u} \tag{4.62}$$

which can be written as

$$\Delta\lambda = \mathbf{v}^T \left(-\mathbf{S}^H \mathbf{\Delta C V} - \mathbf{V}^H \mathbf{\Delta C}^H \mathbf{S}\right) \mathbf{u}. \tag{4.63}$$

Since $\mathbf{\Delta C}$ is a real matrix, $\mathbf{\Delta C}^H = \mathbf{\Delta C}^T$

$$\begin{aligned} \Delta\lambda &= \mathbf{v}^T \left(-\mathbf{S}^H \mathbf{\Delta C V} - \mathbf{V}^H \mathbf{\Delta C}^T \mathbf{S}\right) \mathbf{u} \\ &= -\mathbf{v}^T \mathbf{S}^H \mathbf{\Delta C V u} - \mathbf{v}^T \mathbf{V}^H \mathbf{\Delta C}^T \mathbf{S u}. \end{aligned} \tag{4.64}$$

Next we invoke an identity of the Kronecker product $\otimes$ which states that for a given matrix $\mathbf{Y}$, $\mathbf{A}$, $\mathbf{X}$, and $\mathbf{B}$ [44]

$$\mathbf{Y} = \mathbf{AXB} \quad \Leftrightarrow \quad vec(\mathbf{Y}) = (\mathbf{B}^T \otimes \mathbf{A}) vec(\mathbf{X}) \tag{4.65}$$

$$\mathbf{Y} = \mathbf{AXB} \quad \Leftrightarrow \quad wec(\mathbf{Y}) = (\mathbf{A} \otimes \mathbf{B}^T) wec(\mathbf{X}) \tag{4.66}$$

where $vec(.)$ denotes the vectorization of the matrix (.) formed by column-ordering the matrix (.) into a single column vector and $wec(.)$ denotes the vectorization of the matrix (.) formed by row-ordering the matrix (.) into a single column vector. Applying (4.65) and (4.66), along with the fact that $\Delta\lambda$ is a scalar on (4.64), results in

$$\Delta\lambda = -\left((\mathbf{Vu})^T \otimes \mathbf{v}^T\mathbf{S}^H\right) vec(\mathbf{\Delta C}) - \left(\mathbf{v}^T\mathbf{V}^H \otimes (\mathbf{Su})^T\right) wec(\mathbf{\Delta C}^T). \tag{4.67}$$

Since

$$wec(\mathbf{\Delta C}^T) = vec(\mathbf{\Delta C}) \tag{4.68}$$

we have

$$\begin{aligned} \Delta\lambda &= -\left((\mathbf{Vu})^T \otimes \mathbf{v}^T\mathbf{S}^H\right) vec(\mathbf{\Delta C}) - \left(\mathbf{v}^T\mathbf{V}^H \otimes (\mathbf{Su})^T\right) vec(\mathbf{\Delta C}) \\ &= -\left[\left((\mathbf{Vu})^T \otimes \mathbf{v}^T\mathbf{S}^H\right) + \left(\mathbf{v}^T\mathbf{V}^H \otimes (\mathbf{Su})^T\right)\right] vec(\mathbf{\Delta C}) \end{aligned} \tag{4.69}$$

which has the form

$$\Delta\lambda = \mathbf{g} \cdot vec(\mathbf{\Delta C}) \tag{4.70}$$

with

$$\mathbf{g} = -\left[\left((\mathbf{Vu})^T \otimes \mathbf{v}^T\mathbf{S}^H\right) + \left(\mathbf{v}^T\mathbf{V}^H \otimes (\mathbf{Su})^T\right)\right] \tag{4.71}$$

which can be shown to be a row vector. Thus, for a passivity violation at a particular frequency, (4.70) provides the means for restoring passivity at that point.

When a passivity violation band is detected, (4.70) is applied to the point of maximum violation which was obtained from (4.47). For cases where there are more than one violation band, passivity compensation can be done simultaneously for all violation bands by setting up (4.70) for each band, resulting in a set of least-squares equations in the form of

$$\Delta\lambda = \mathbf{G} \cdot vec(\mathbf{\Delta C}) \tag{4.72}$$

where $\Delta\lambda$ is a vector formed by the magnitudes of maximum violations in each band and $\mathbf{G}$ is a matrix consisting of several rows of $\mathbf{g}$'s.

In order to retain the accuracy of the model, we minimize the change in the scattering matrix as passivity enforcement is carried out. To do this, we return to (4.52) which defines the change in the scattering matrix after passivity compensation and relate that to the perturbation of the residues $\mathbf{\Delta C}$. It can be shown that [45]

$$\|\mathbf{\Delta S}\|^2 = trace(\mathbf{\Delta C P \Delta C}^T) = vec(\mathbf{\Delta C})^T \mathbf{H} vec(\mathbf{\Delta C}) \tag{4.73}$$

where $\mathbf{P}$ is the controllability Grammian obtained by solving the Lyapunov equation

$$\mathbf{AP} + \mathbf{PA}^H + \mathbf{BB}^H = \mathbf{0} \tag{4.74}$$

and $\mathbf{H}$ is a matrix formed by stacking $\mathbf{P}$ on the diagonal

$$\mathbf{H} = \begin{bmatrix} \mathbf{P} & 0 & \cdots & 0 \\ 0 & \mathbf{P} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{P} \end{bmatrix}. \tag{4.75}$$

Equations (4.72) and (4.73) together result in an optimization problem which can be solved iteratively to satisfy passivity while minimizing the change in the response. Since the objective function given in (4.73) is quadratic in nature, the problem is solved by utilizing a quadratic programming routine where the overall problem is

$$\min \Big(vec(\mathbf{\Delta C})^T \mathbf{H} vec(\mathbf{\Delta C}) \Big) \quad \text{subject to} \quad \Delta\lambda = \mathbf{G} \cdot vec(\mathbf{\Delta C}). \tag{4.76}$$

4.2.2.3 Summary

The overall process of passivity enforcement is summarized in the flowchart given in Fig. 4.3.

4.2.3 Recursive Convolution

In this section, the recursive convolution algorithm which is used to obtain the time-domain response of the system will be introduced. Assume that the methods in the previous section have been applied to the system to generate a stable, passive and proper rational function approximation, where the transfer function $H(s)$ is given by

$$H(s) = \sum_{n=1}^{N} \frac{c_n}{s + a_n} + d. \tag{4.77}$$

Note that for the discussions in this section, the negative sign in the denominator of (4.1) has been absorbed into a_n in (4.77). For this transfer function, its input-output relationship is given by

$$Y(s) = H(s) \cdot X(s) \tag{4.78}$$

where $X(s)$ is the input function and $Y(s)$ is the output function. For each term in the summation of (4.77), we have

$$Y_n(s) = H_n(s) \cdot X(s) = \left[\frac{c_n}{s + a_n}\right] \cdot X(s). \tag{4.79}$$

In the time domain this corresponds to

$$y_n(t) = c_n e^{-a_n t} * x(t) \tag{4.80}$$

where $*$ denotes convolution. Equation (4.80) can be evaluated most effectively using the recursive convolution [46] method which will be presented next.

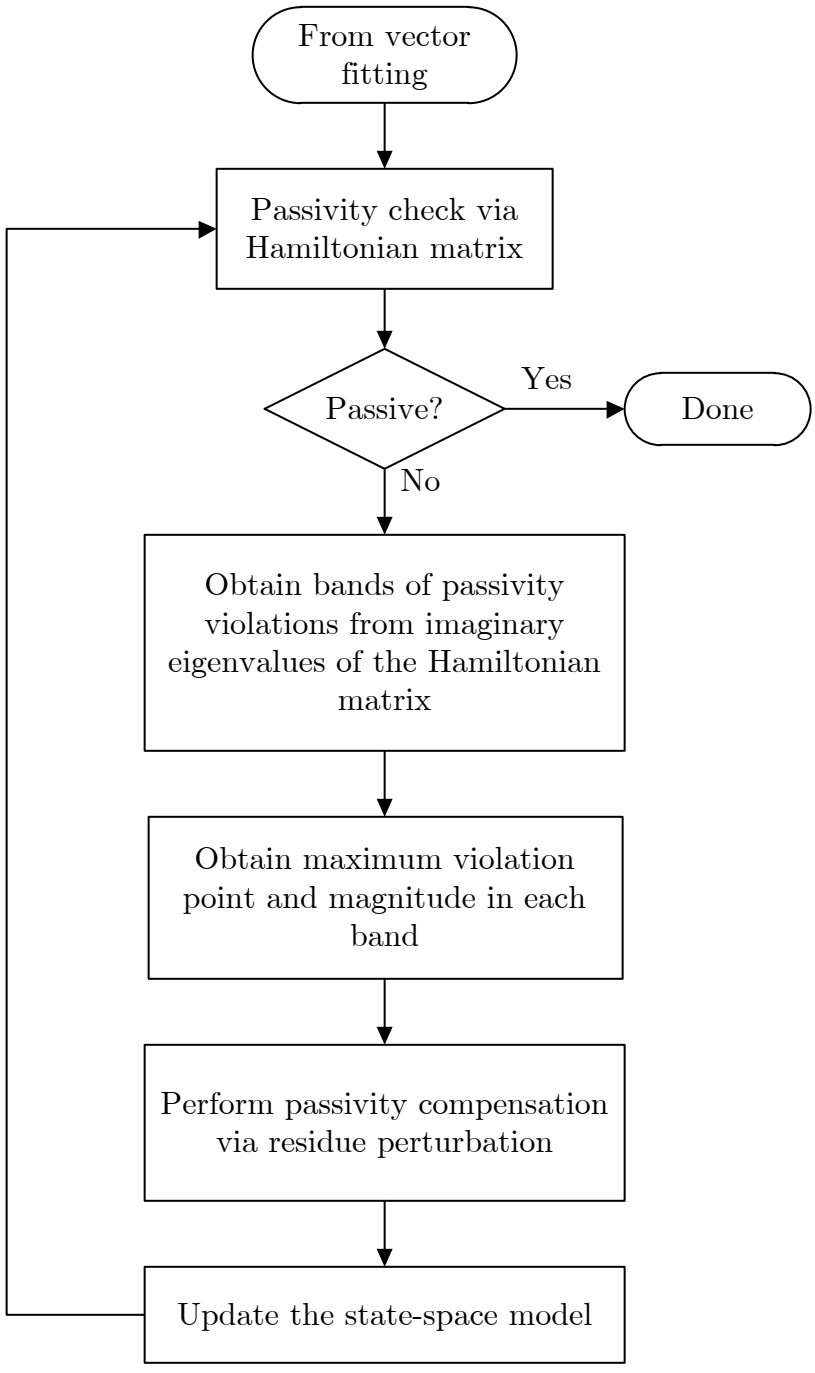


Figure 4.3: Flowchart of the passivity enforcement process.

The goal is to evaluate the function

$$y(t) = Ae^{-\alpha t} * x(t). \tag{4.81}$$

This is equivalent to

$$y(t) = \int_0^t Ae^{-\alpha\tau}x(t-\tau)d\tau = \int_0^h Ae^{-\alpha\tau}x(t-\tau)d\tau + \int_h^t Ae^{-\alpha\tau}x(t-\tau)d\tau. \tag{4.82}$$

Assuming a step invariant (constant) behavior of the input function, the first integral can be written as

$$\int_0^h Ae^{-\alpha\tau}x(t-\tau)d\tau = Ax(t-h)\int_0^h e^{-\alpha\tau}d\tau \tag{4.83}$$

which can be evaluated to yield

$$\int_0^h Ae^{-\alpha\tau}x(t-\tau)d\tau = \frac{Ax(t-h)}{\alpha}\left(1-e^{-\alpha h}\right). \tag{4.84}$$

Setting $\tau = \tau' + h$ in the second integral yields

$$\begin{aligned} \int_h^t Ae^{-\alpha\tau}x(t-\tau)d\tau &= \int_0^{t-h} Ae^{-\alpha(\tau'+h)}x(t-\tau'-h)d\tau' \\ &= e^{-\alpha h}\int_0^{t-h} Ae^{-\alpha\tau'}x(t-\tau'-h)d\tau' \\ &= e^{-\alpha h}y(t-h). \end{aligned} \tag{4.85}$$

Thus the overall result is then

$$y(t) = \frac{Ax(t-h)}{\alpha}\left(1-e^{-\alpha h}\right) + e^{-\alpha h}y(t-h) \tag{4.86}$$

which is the general recursive convolution formula for a step-invariant approximation.

Returning to (4.80) and applying the result obtained in (4.86) with a time step $T = h$ gives

$$y_n(t) = e^{-a_n T} y_n(t-T) + \frac{c_n}{a_n} x(t-T)\left(1 - e^{-a_n T}\right). \tag{4.87}$$

Therefore the complete solution at each time step is given by

$$y(t) = d \cdot x(t-T) + \sum_{n=1}^{N} y_n(t) \tag{4.88}$$

where $y_n(t)$ is as given in (4.87).

Before concluding this section, let us examine a special case when the poles and residues appear in complex conjugate pairs. In that instance, (4.87) would yield

$$y_n(t) = e^{-a_n T} y_n(t-T) + \frac{c_n}{a_n} x(t-T)\left(1 - e^{-a_n T}\right) \tag{4.89}$$

$$y_{n+1}(t) = e^{-a_n^* T} y_{n+1}(t-T) + \frac{c_n^*}{a_n^*} x(t-T)\left(1 - e^{-a_n^* T}\right) \tag{4.90}$$

where the asterisk "*" indicates complex conjugacy. It can be shown that $y_{n+1}(t) = y_n^*(t)$. Therefore, this leads to

$$y_n(t) = e^{-a_n T} y_n(t-T) + \frac{c_n}{a_n} x(t-T)\left(1 - e^{-a_n T}\right) \tag{4.91}$$

$$y_{n+1}(t) = e^{-a_n^* T} y_n^*(t-T) + \frac{c_n^*}{a_n^*} x(t-T)\left(1 - e^{-a_n^* T}\right) \tag{4.92}$$

Examining (4.91) and (4.92) reveals that $y_n(t) + y_{n+1}(t)$ results in a real quantity. Thus the properties of a real system are preserved by ensuring that each complex pole appears along with its complex conjugate and that the residues corresponding to those poles also come in complex conjugate pairs.

Table 4.1: RMS error of the model before and after passivity enforcement.

	S_{11}	S_{12}	S_{21}	S_{22}
From VFIT	0.00727	0.0112	0.0139	0.00720
After passivity enforcement	0.00449	0.00435	0.00432	0.00423

4.2.4 Example

In this section, we present an example to illustrate the vector fitting, passivity enforcement and recursive convolution processes. The scattering parameters of a two-port interconnect structure are obtained in the frequency range of 50 MHz – 5 GHz. The vector fitting method is used to obtain a model for the system, fitting all the elements of the two-port system using the same set of poles with an order of 40. Two vector fitting iterations are used, which take a total of 1.16 s as measured on a desktop computer with an AMD 2.3 GHz Dual Core processor and 1 GB of RAM. The passivity of the system was analyzed and the Hamiltonian matrix revealed two passivity violation regions. Passivity enforcement was carried out which converged after four iterations, lasting an additional 0.83 s. Plots of all the S-parameters are shown in Figs. 4.4 – 4.7. Table 4.1 shows the root-mean-square (RMS) error of the model compared to the original signal before and after passivity enforcement. We see that the overall accuracy of the model is retained throughout the process. A plot of the eigenvalues of the dissipation matrix is shown in Fig. 4.8, verifying the passivity compensation process. A time-domain simulation is done by utilizing the recursive convolution process with the model developed. A single pulse with rise and fall time of 1 ns and with a pulse width of 8 ns is sent at port 1 and the responses at both ports were evaluated. Owing to the numerical superiority of the recursive convolution process, this took only 0.062 s. The result is shown in Fig. 4.9.

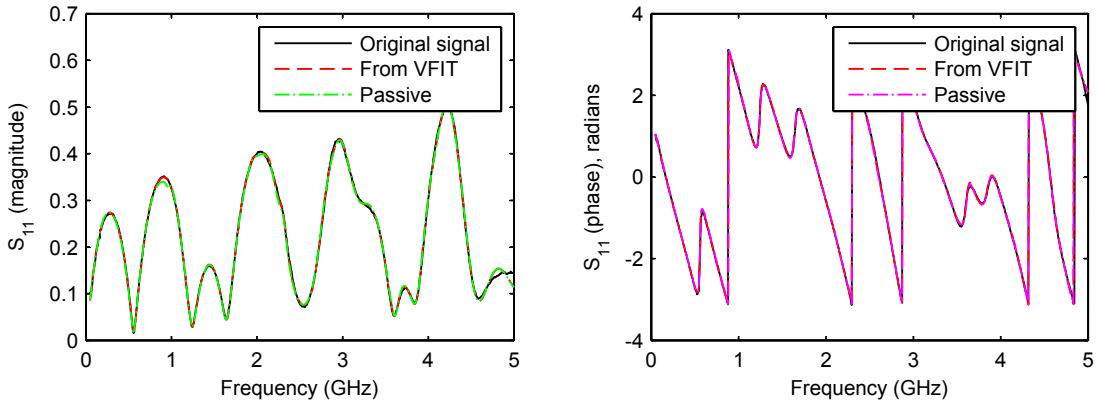


Figure 4.4: Comparison of S_{11} of the measured data and the model.

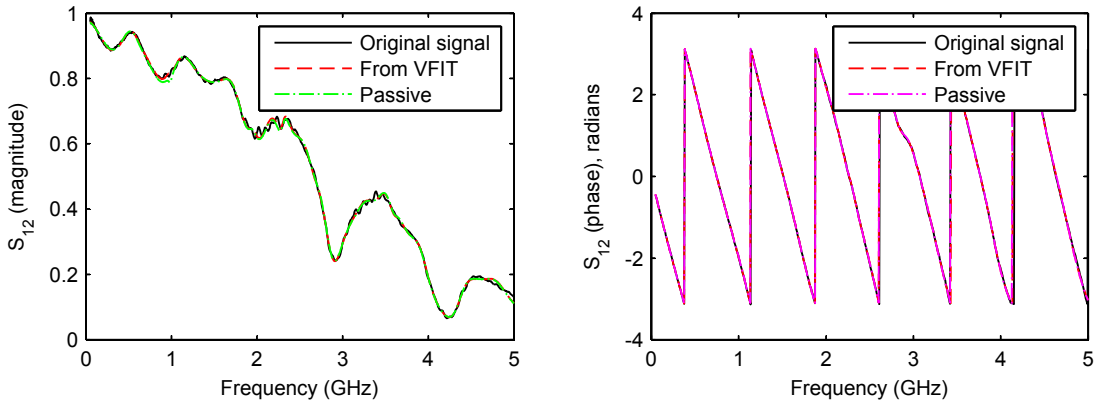


Figure 4.5: Comparison of S_{12} of the measured data and the model.

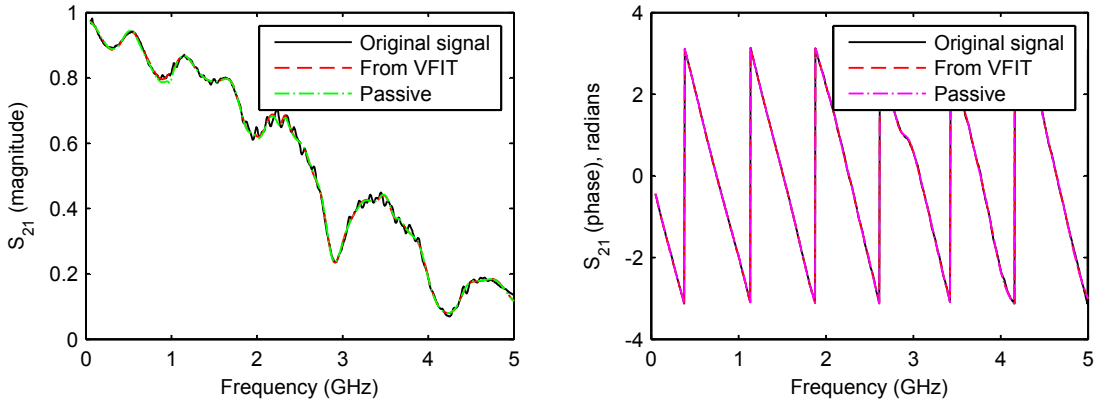


Figure 4.6: Comparison of S_{21} of the measured data and the model.

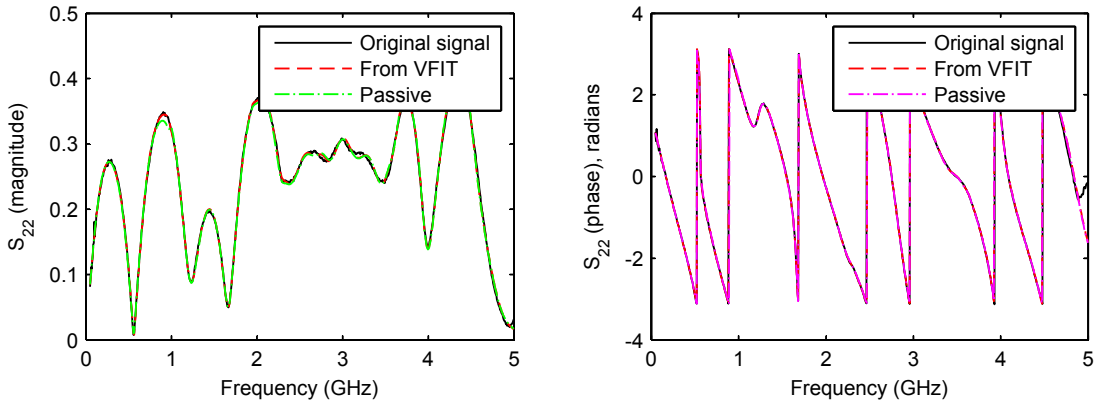


Figure 4.7: Comparison of S_{22} of the measured data and the model.

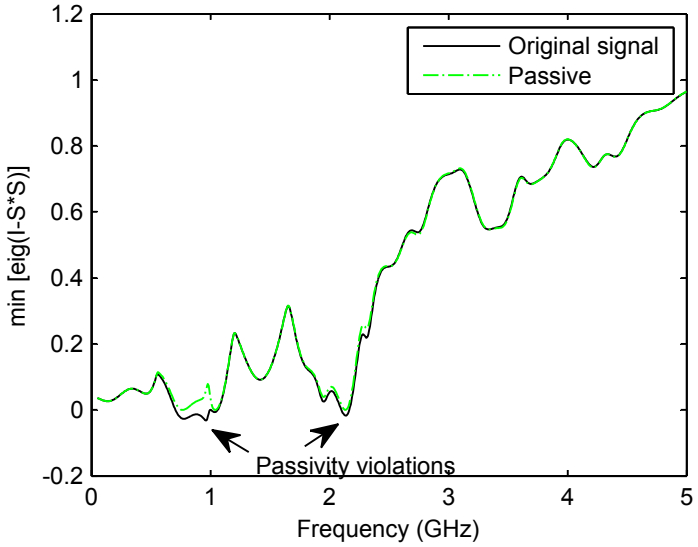


Figure 4.8: Eigenvalues of the dissipation matrix. Negative values indicate passivity violation.

4.3 S-Parameter Fast Convolution

In this section, we will describe a fast convolution based approach for the incorporation of blackbox macromodels in circuit simulators. We begin with an overview of the general convolution process. Consider a blackbox respresented by its n-port scattering parameters, $S(\omega)$. The response at the terminals of the blackbox is given

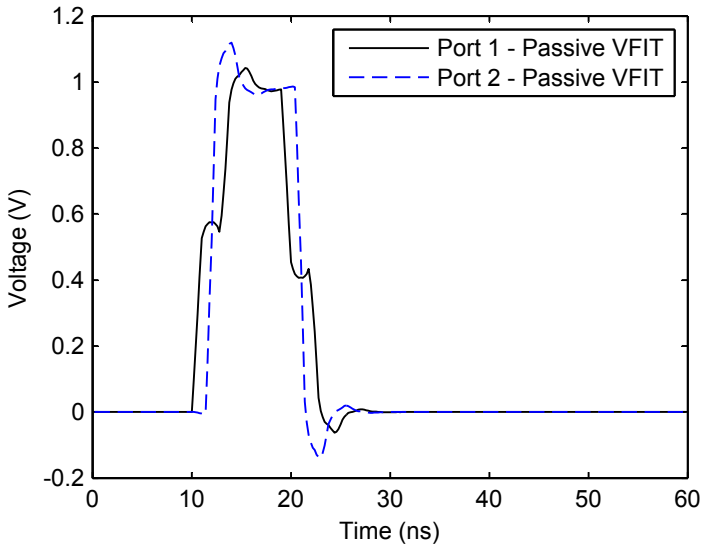


Figure 4.9: Time-domain response.

by

$$B(\omega) = S(\omega)A(\omega) \tag{4.93}$$

where $B(\omega)$ and $A(\omega)$ are the reflected and incident waves respectively. In the time domain, this becomes

$$b(t) = s(t) * a(t) \tag{4.94}$$

where $*$ indicates the convolution operator given by

$$s(t) * a(t) = \int_{-\infty}^{\infty} s(t-\tau)a(\tau)d\tau. \tag{4.95}$$

When the time variable is discretized, this convolution becomes

$$s(t) * a(t) = s(0)a(M)\Delta t + \sum_{k=1}^{M} s(k)a(M-k)\Delta t \tag{4.96}$$

where Δt is the time step and M is the index associated with the current time. With this formulation, (4.94) can be written as

$$b(t) = s_o a(t) + h(t) \tag{4.97}$$

where $s_o = s(0)\Delta t$ and $h(t)$ is the history of the scattered voltage wave given by

$$h(t) = \sum_{k=1}^{M} s(k)a(M-k)\Delta t. \tag{4.98}$$

4.3.1 Fast Convolution Using δ-Function Convolution

Most of the computational burden in the time-domain simulation rests on the calculation of $h(t)$ in (4.98) which involves an expensive convolution operation, where the computational complexity is known to be $O(n^2)$ where n is the number of sample points. Our approach to alleviate this problem takes advantage of the fact that scattering parameter impulse responses have relatively short durations and consist of pulses that decay very rapidly with time. A close observation of the time-domain scattering parameter data generated by the IFFT shows that the vast majority of points have small magnitude and consequently can be neglected. For instance, the insertion loss scattering parameter of a microstrip line was measured on a network analyzer up to 40 GHz. When the data is processed through a 801-point IFFT, only 25 points of the resulting time-domain sequence are larger than 1% of the maximum (absolute) value. This can also be easily observed by looking at the plots of the impulse responses shown in Fig. 4.10.

Consequently, most of the $s(k)$ terms in the summation in (4.98) will be zeros and the calculation of $h(t)$ can be accelerated dramatically. As a reformulation, we can assume that the discrete frequency-domain scattering parameter transfer functions

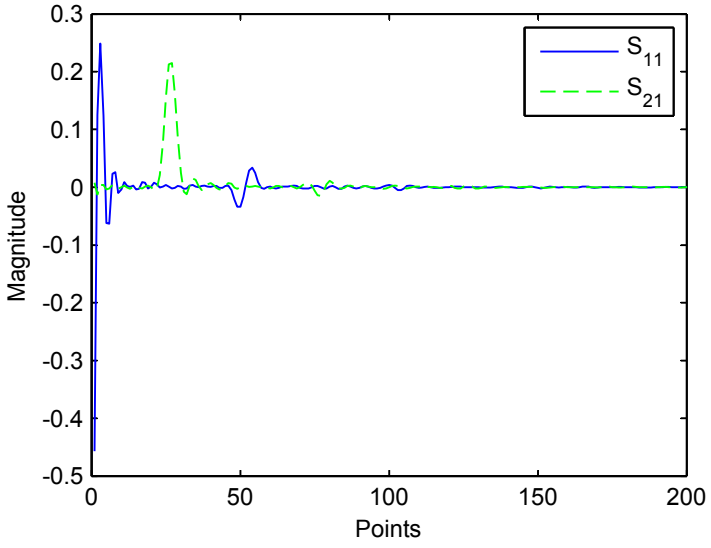


Figure 4.10: Time-domain scattering parameter responses for a microstrip showing the rapid decay of the function. Only the first 200 points from the IFFT are shown.

can be described in the form

$$S_d(q) = \sum_{k=1}^{L} c_k e^{j2\pi qk} \tag{4.99}$$

in which the c_k's and k's are parameters to be determined. L is the order of the approximation that satisfies $L << N$ where N is the total number of simulation points. With this representation, the associated time-domain function takes the form of a train of impulses whose weights are given by the c_k's:

$$s_d(p) = \sum_{k=1}^{L} c_k \delta(p-k). \tag{4.100}$$

Convolution with an excitation function $a_d(p)$ then gives

$$h_d(p) = \left[\sum_{k=1}^{L} c_k \delta(p-k)\right] * a_d(p) = \sum_{k=1}^{L} c_k a_d(p-k). \tag{4.101}$$

In a typical approach, the c_k's are obtained by taking the inverse discrete Fourier transform or IFFT of the frequency-domain transfer function. If the transfer functions are scattering parameters, most of these c_k's will be negligibly small and thus only a few (L) will need to be retained for the representation described in (4.100). In addition, when the reference system is optimally chosen, the time-domain scattering parameters die out quickly, leading to even fewer points in the delta-function sequence. In general, the choice of L is directly predicated by the desired accuracy and is also a strong function of the frequency-domain data.

4.3.2 DC Extraction and Causality Enforcement

The IFFT process used in the fast convolution method requires input data down to DC in the frequency domain in order to generate a reliable result. However, most data used for blackbox macromodels are obtained either from network analyzer measurements or full-wave electromagnetic solvers, neither of which operates well at low frequencies. Consequently, most frequency-domain data are often missing the low frequency and DC values, and they must be extracted from the available data. We present two methods that can be used to extrapolate the given data down to DC.

In the first method, the Smith chart is used to extrapolate the data. On the Smith chart, S-parameters follow the general pattern of growing or decaying clockwise-moving spirals with increasing frequency. Moreover, at DC, the S-parameters of a physical circuit value must be real and must lie on the horizontal axis of the Smith chart. With these considerations, we can assume a mathematical behavior described by

$$S(f) = r_0 e^{j\theta} + r e^{\pm f\alpha} e^{-j2\pi f\tau} \tag{4.102}$$

for the low-frequency behavior of an S-parameter on the Smith chart. An algorithm can be devised to extract the values of r_0, r and τ using data points from the lowest

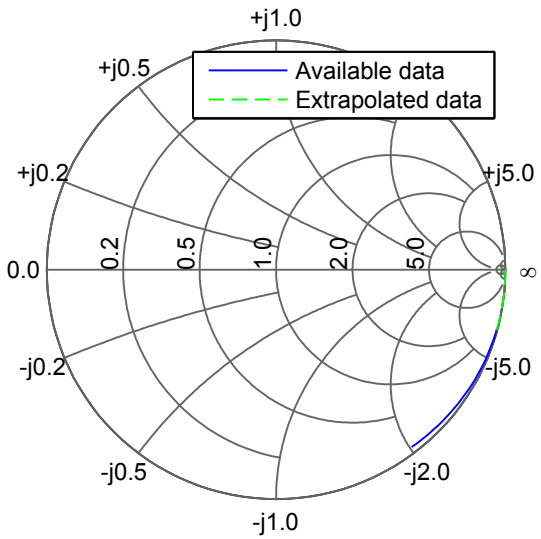


Figure 4.11: Example of DC extraction on the Smith chart.

frequencies [47]. Extrapolation of values for frequencies down to DC can then be achieved. This method is illustrated in Fig. 4.11.

A second possible method is to use the vector fitting process described in Section 4.2.1 on the low frequency values of the available data. The model is then used to generate missing data down to DC. Note that in this process, the computational time is relatively small compared to the generation of a full model over the entire frequency range as the process is only applied to the low frequency values (typically the first 10-30 points) of the data and the order is very small (typically 1-3). Furthermore, since the model is then used to generate discrete data, passivity enforcement can be done relatively easily by checking condition (4.49) at the extrapolated points.

Regardless of which method is used for the DC extraction process, when data points are artificially added into actual data, one must ensure that the physical properties of the system are not altered. In particular, time-domain signals associated with physical systems must be causal [48]. This means that the response to an excitation

starting at $t = 0$ must be null for $t < 0$:

$$h(t) = 0, \quad t < 0 \tag{4.103}$$

where $h(t)$ is the response of a system due to an excitation starting at $t = 0$. The response $h(t)$ can be considered as the superposition of an even and an odd function defined as

$$h_e(t) = \frac{1}{2}\left[h(t) + h(-t)\right] \qquad \text{even function} \tag{4.104}$$

$$h_o(t) = \frac{1}{2}\left[h(t) - h(-t)\right] \qquad \text{odd function.} \tag{4.105}$$

If $h(t)$ is a causal function, then

$$h_o(t) = \begin{cases} h_e(t), & t > 0 \\ -h_e(t), & t < 0 \end{cases} \tag{4.106}$$

or

$$h_o(t) = \text{sgn}(t) h_e(t). \tag{4.107}$$

Therefore, $h(t)$ can be rewritten as:

$$h(t) = h_e(t) + \text{sgn}(t) h_e(t). \tag{4.108}$$

Causality in the time-domain data is then enforced by carrying out the following steps. First the real part of the frequency-domain data is inverted into the time domain via IFFT which yields the even part of the time-domain response. Next, the full time-domain response is generated from the even part using (4.108). This illustrates that the time-domain response can be generated entirely from the real part of the frequency-domain data [48].

4.3.3 Example

In this section, we present an example to illustrate the fast convolution approach. The scattering parameters of a two-port interconnect structure are obtained in the frequency range of 50 MHz – 5 GHz. Note that this is the same example used in Section 4.2.4. Since the original data is only specified down to 50 MHz, the DC extraction process explained in the previous section is used to generate the missing data. The result for S_{11} is shown in Fig. 4.12. Next, a causal IFFT routine is used to generate the impulse response. The result for S_{11} is again shown in Fig. 4.13. As expected, most of the points have small magnitudes and can be neglected. Specifically, for S_{11}, out of the 801 points, only 123 points have magnitudes larger than 0.001 of the maximum (absolute) value. Next, the fast δ-function convolution explained in Section 4.3.1 is used to generate the time-domain response. A single pulse with rise and fall time of 1 ns and with a pulse width of 8 ns is sent at port 1, and the responses at both ports are evaluated. The result is shown in Fig. 4.14. Comparing this to Fig. 4.9 in Section 4.2.4, we see that both methods, the passive MOR via vector fitting and fast convolution, generate similar results. However, the overall process for the fast convolution approach takes a mere 0.125 s compared to 2.052 s for the MOR approach. Both simulations were performed on a desktop computer with an AMD 2.3 GHz Dual Core processor and 1 GB of RAM. A detailed comparative study of the two methods will be performed in Section 4.4.

Before concluding this section, we illustrate the importance of the DC extraction process using this example. Fig. 4.15 shows the time-domain responses that were generated from the same data but without DC extraction. A substantial loss in accuracy is observed.

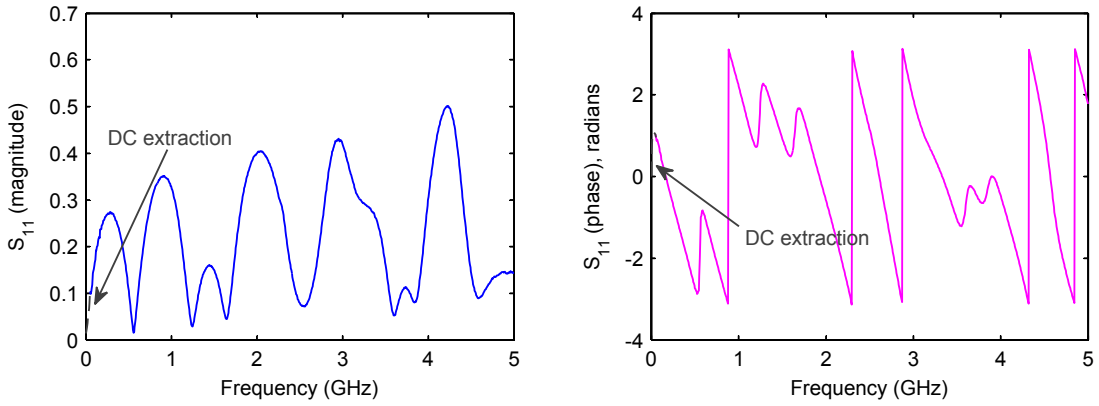


Figure 4.12: Example of DC extraction process on S_{11}.

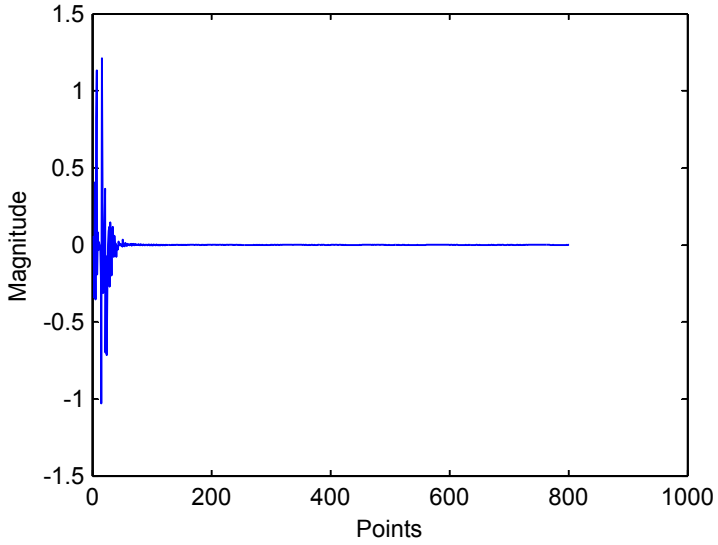


Figure 4.13: Impulse response of S_{11} showing the rapid decay of the function.

4.4 A Comparative Study of MOR via Vector Fitting and Fast Convolution

In this section, we present a comparative study [49] of the two techniques for macromodel generation presented in the previous sections. MOR via vector fitting and fast convolution will be compared in terms of computational speed and accuracy. Two computer programs were written in C++ using the Visual Studio environment.

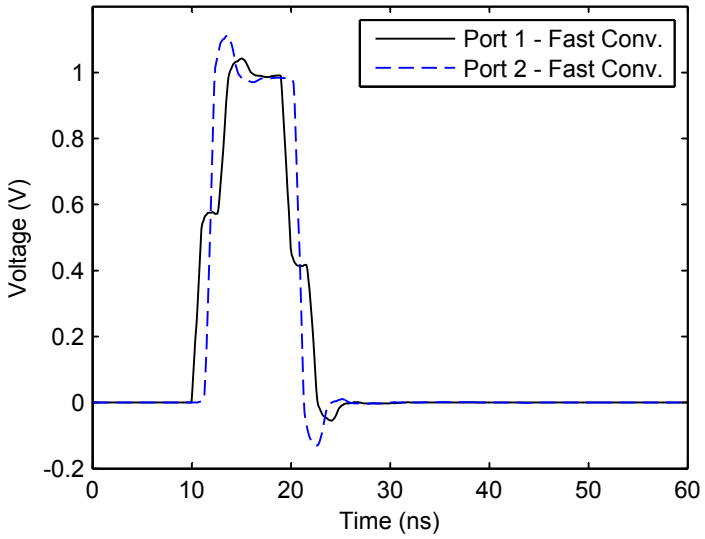


Figure 4.14: Time-domain response using the fast δ-function convolution.

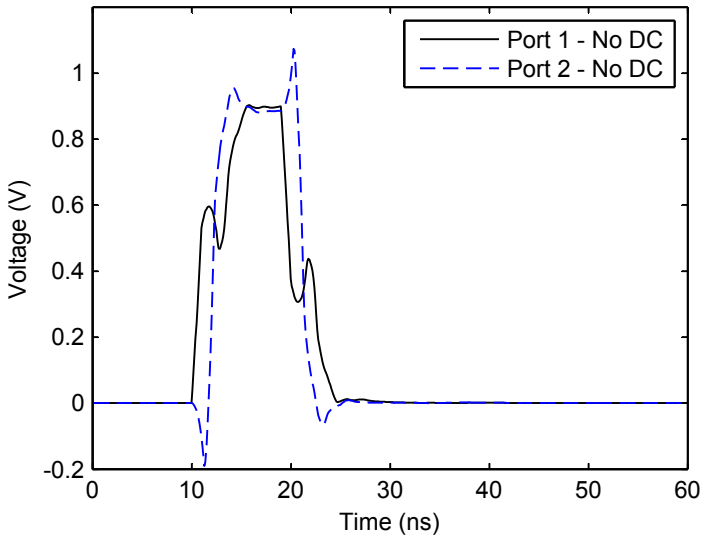


Figure 4.15: Time-domain response using the fast δ-function convolution without DC extraction.

Table 4.2: Data file descriptions.

Name	Ports	Description
Bbx-1	2	2-port I/O network: 1 MHz – 5 GHz
Bbx-2	2	Microstrip line: 50 MHz – 7 GHz
Bbx-3	2	Microstrip coupler: 50 MHz – 5 GHz
Bbx-4	2	Microstrip on FR4: 2 GHz – 50 GHz
Bbx-5	2	Lossy microstrip line: 0 – 20 GHz
Bbx-6	2	18-inch meander: 300 KHz – 6 GHz
Bbx-7	4	4-port data A: 50 MHz – 20.05 GHz
Bbx-8	4	4-port data B: 0 – 50 GHz
Bbx-9	4	4-port data C: 50 MHz – 20.05 GHz

For the program utilizing MOR via vector fitting, the speed of the pole identification process was further enhanced using the method described in [11]. Various two-port and four-port networks were used as the benchmarks. A description of these data files is given in Table 4.2. The files were obtained either from network analyzer measurements or from full-wave electromagnetic field solvers.

The excitation used was a trapezoidal pulse with an amplitude of 1 V, rise and fall times of 1 ns and a pulse width of 8 ns. The simulation time was 50 ns. In all cases, the excitation was provided at port 1, using a generator with a 50 Ω internal impedance. All the other ports were left open.

Table 4.3 shows a comparison of the runtime between the two methods. In the MOR case, the simulation times for vector fitting, passivity enforcement and recursive convolution are shown separately. In some cases, fittings with different orders are shown to illustrate the effects on speed. All of the simulations were performed on a desktop computer with an AMD 2.3 GHz Dual Core processor and 1 GB of RAM.

For the MOR based method, the order of the approximation, N, in the partial fraction expansion is critical in determining the speed and accuracy of the simulation results. The computational advantage gained through the use of recursive convolution may be lost if the order is high. In addition, it was observed that for the most part, passivity enforcement contributed to the largest computational time and represented

Table 4.3: Benchmark for MOR and fast convolution techniques.

Data file	No. of points	MOR with Vector Fitting					Fast Conv.
		Order	Time (s)				Time (s)
			VFIT†	Passivity Enforc.	Recursive Conv‡	TOTAL	
Bbx-1	501	10*	0.14	0.01nv	0.02	0.17	0.078
Bbx-2	801	20*	0.41	5.47	0.03	5.91	0.110
Bbx-3	801	40*	1.08	0.08nv	0.06	1.22	0.125
Bbx-4	801	60*	2.25	1.89	0.09	4.23	0.125
		100	3.17	5.34	0.16	8.67	
Bbx-5	2001	50*	4.97	0.09nv	0.28	5.34	0.328
Bbx-6	801	100*	3.17	0.56nv	0.16	3.89	0.109
Bbx-7	1601	100*	24.59	28.33	1.31	53.23	0.438
		120	31.16	27.64	1.58	60.38	
Bbx-8	5096	220	250.08	25.77nv	10.05	285.90	2.687
Bbx-9	1601	200*	58.47	91.63	2.59	152.69	0.469
		250	80.64	122.83	3.22	206.69	
		300	106.53	61.58nv	3.86	171.97	

* Lowest order for a visually good fit, rounded up to the nearest 10.
† 2 iterations of VFIT done.
‡ Time for one simulation up to $Num\_time = 2 \times Num\_freq$. Time step was chosen such that the total simulation time is 50 ns.
nv No passivity violation. No passivity enforcement necessary. Time needed was to check passivity.

an important fraction of the total time. This is due to the process of iteratively determining the eigenvalues of the Hamiltonian and perturbing the residue matrix for passivity enforcement.

The results summarized in Table 4.3 as well as numerous additional simulations permit us to conclude that an MOR-based technique for blackbox macromodeling does not offer a significant advantage over convolution-based methods. In fact, if scattering parameters are used, fast convolution can be used to accelerate the simulation. All simulations indicate that reliable and comparable accuracy can be obtained by

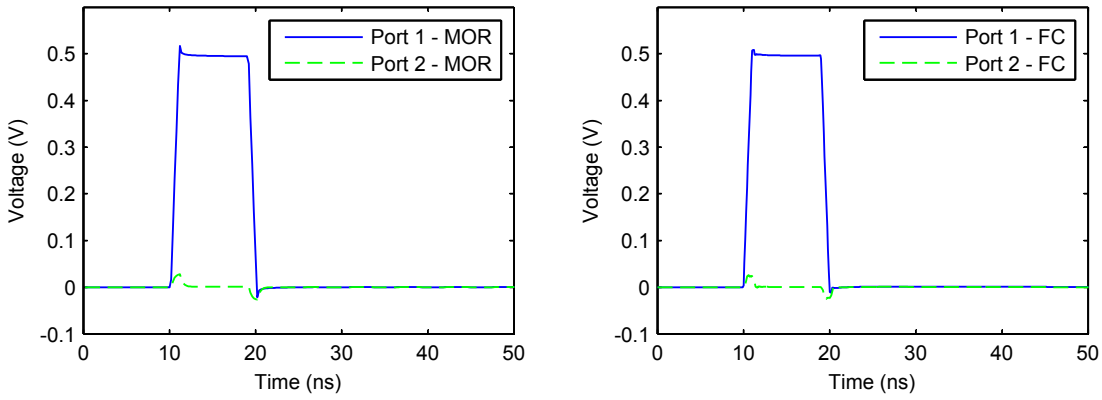


Figure 4.16: Simulation comparisons for MOR and fast convolution for Bbx-1. Left: Passive MOR. Right: Fast convolution.

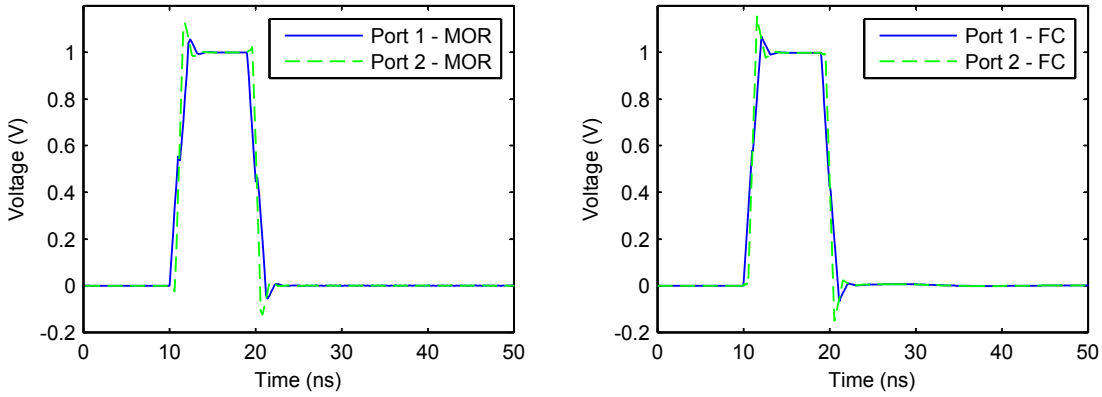


Figure 4.17: Simulation comparisons for MOR and fast convolution for Bbx-2. Left: Passive MOR. Right: Fast convolution.

both convolution and MOR techniques. Plots of the voltage waveforms at the inputs and outputs for Bbx-1 and Bbx-2 are shown in Figs. 4.16 – 4.17 for comparison.

4.5 Integrating Blackbox in LIM

We conclude this chapter by showing how blackbox macromodels developed in the previous sections can be incorporated into a LIM simulation [50]. From the definition of scattering parameters, the incident and reflected waves can then be related to the

voltage and currents at the terminals using

$$\mathbf{a}(t) = \frac{1}{2}\left[\mathbf{v}(t) + \mathbf{Z_o}\mathbf{i}(t)\right] \tag{4.109}$$

$$\mathbf{b}(t) = \frac{1}{2}\left[\mathbf{v}(t) - \mathbf{Z_o}\mathbf{i}(t)\right] \tag{4.110}$$

where $\mathbf{a}(t)$ and $\mathbf{b}(t)$ are the incident and reflected waves respectively, $\mathbf{v}(t)$ and $\mathbf{i}(t)$ are the terminal voltage and current vectors respectively and $\mathbf{Z_o}$ is the reference impedance matrix. Substituting (4.109) and (4.110) into (4.88) for blackboxes represented by MOR via vector fitting and (4.97) for blackboxes represented by the fast convolution method yields

$$\frac{1}{2}\left[\mathbf{v}(t) - \mathbf{Z_o}\mathbf{i}(t)\right] = \frac{1}{2}\mathbf{s_o}\left[\mathbf{v}(t) + \mathbf{Z_o}\mathbf{i}(t)\right] + \mathbf{h}(t) \tag{4.111}$$

from which $\mathbf{i}(t)$ can be solved as

$$\mathbf{i}(t) = \mathbf{Z_o^{-1}}[\mathbf{1}+\mathbf{s_o}]^{-1}\left[\mathbf{1}-\mathbf{s_o}\right]\mathbf{v}(t) - 2\mathbf{Z_o^{-1}}[\mathbf{1}+\mathbf{s_o}]^{-1}\mathbf{h}(t). \tag{4.112}$$

In most circuit simulators, the voltage-current relationship is expressed in the form of stamp parameters that represent subnetwork components. In this form, (4.112) can be simplified to read

$$\mathbf{i}(t) = \mathbf{Y_{stamp}}\mathbf{v}(t) - \mathbf{i_{stamp}} \tag{4.113}$$

where

$$\mathbf{Y_{stamp}} = \mathbf{Z_o^{-1}}[\mathbf{1} + \mathbf{s_o}]^{-1}\left[\mathbf{1} - \mathbf{s_o}\right] \tag{4.114}$$

and

$$\mathbf{i_{stamp}} = 2\mathbf{Z_o^{-1}}[\mathbf{1} + \mathbf{s_o}]^{-1}\mathbf{h}(t). \tag{4.115}$$

The definitions of $\mathbf{s_o}$ and $\mathbf{h}(t)$ differ depending on the representation of the blackbox.

With this formulation, the blackbox can then be incorporated into a LIM simulation where the blackbox is represented by the currents through its terminals. At each time step, the currents into these branches are calculated using (4.113); next the currents at all the external branches are evaluated using the LIM updating equation given in (2.4). Finally, all the nodes voltages are updated using (2.2). The algorithm is summarized as follows:

Algorithm 1 LIM simulation with blackbox models

```
for time = 1 to N_time do
  for blackbox = 1 to N_b-box do
    Calculate macromodel branch currents using (4.113)
  end for
  for branch = 1 to N_branch do
    Update current as per (2.4)
  end for
  for node = 1 to N_node do
    Update voltage as per (2.2)
  end for
end for
```

We present an example to verify the method. Consider the circuit shown in Fig. 4.18. The two-port blackbox consists of scattering parameter data of an interconnect network measured from 2 GHz – 5 GHz with 801 frequency points. The MOR method via vector fitting is used to generate a pole-residue approximation of the system with an order of 60. The entire circuit is then simulated using LIM. The excitation is provided by a current source pulse connected at node 1. The magnitude of the pulse is 20 mA with rise and fall times of 0.1 ns and pulse width of 2 ns. The resulting transient response waveforms are shown in Fig. 4.19 for the voltage at nodes 1 and 4. Comparison with simulations using Agilent's Advanced Designed Systems (ADS) [51] shows small differences between the two methods which are attributed mainly to the inaccuracies in the model.

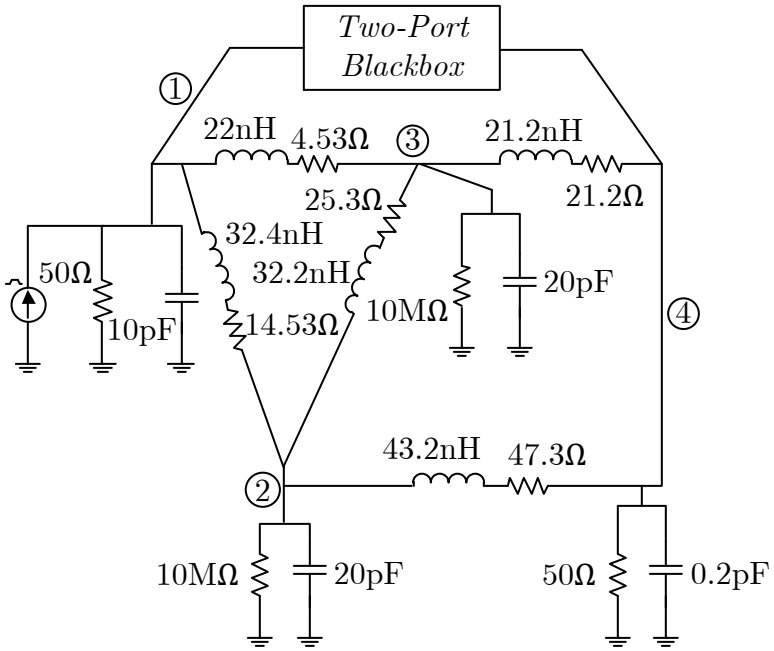


Figure 4.18: Example circuit containing a blackbox model.

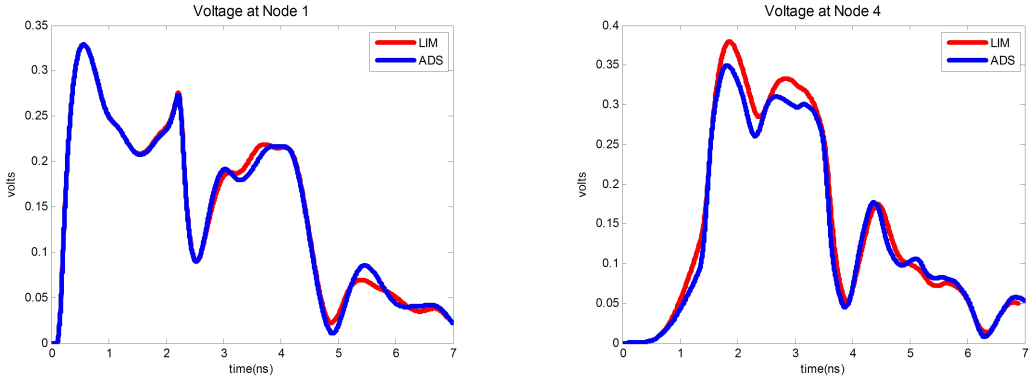


Figure 4.19: Simulated voltage waveforms for nodes 1 and 4 of the circuit in Fig. 4.18.

CHAPTER 5

CMOS CIRCUIT SIMULATION IN LIM

5.1 Introduction

For signal integrity analysis, a majority of the circuits being analyzed consist of linear, passive interconnects and nonlinear drivers at the terminals. In the preceding chapters, we have seen how linear devices such as resistors, capacitors, inductors and even blackbox macromodels which are characterized in the frequency domain can be handled and included in a LIM simulation. In this chapter, we present the formulation for nonlinear devices. Due to the dominance of CMOS devices in the integrated circuit industry today, we focus our attention mainly on the inclusions of MOSFETs into LIM simulations.

5.2 CMOS Circuit Simulation Using the Shichman-Hodges Model

LIM can be easily applied to CMOS circuits. When a CMOS device is present, the drain current of the CMOS is used as the branch current in place of (2.4). The CMOS drain current can be calculated using the appropriate model for the device. In this work, we adopt the Shichman-Hodges model as used in [52, 53] to model the

CMOS devices where the drain current for an NMOS, I_{Dn}, is given by

$$I_{Dn} = 0, \quad V_G - V_S < V_{Tn}; \quad V_D - V_S \geq 0 \quad \text{(cutoff)}$$

$$\begin{aligned} I_{Dn} &= \frac{K_n W_n}{L_n} \left(V_G - V_S - V_{Tn} - 0.5(V_D - V_S)\right)(V_D - V_S), \\ V_G - V_S &> V_{Tn}; \quad 0 < V_D - V_S < V_G - V_S - V_{Tn} \quad \text{(ohmic)} \end{aligned} \tag{5.1}$$

$$\begin{aligned} I_{Dn} &= \frac{K_n W_n}{2L_n}(V_G - V_S - V_{Tn})^2, \\ V_G - V_S &> V_{Tn}; \quad V_D - V_S > V_G - V_S - V_{Tn} \quad \text{(saturation)} \end{aligned}$$

where K_n, W_n, L_n, and V_{Tn} are the transconductance, channel width, channel length, and threshold voltage for the NMOS device respectively, V_G is the gate voltage, V_S is the source voltage and V_D is the drain voltage. Similarly, the drain current of a PMOS, I_{Dp}, is given by

$$I_{Dp} = 0, \quad V_G - V_S > V_{Tp}; \quad V_D - V_S \leq 0 \quad \text{(cutoff)}$$

$$\begin{aligned} I_{Dp} &= \frac{-K_p W_p}{L_p} \left(V_G - V_S - V_{Tp} - 0.5(V_D - V_S)\right)(V_D - V_S), \\ V_G - V_S &< V_{Tp}; \quad 0 > V_D - V_S > V_G - V_S - V_{Tp} \quad \text{(ohmic)} \end{aligned} \tag{5.2}$$

$$\begin{aligned} I_{Dp} &= \frac{-K_p W_p}{2L_p}(V_G - V_S - V_{Tp})^2, \\ V_G - V_S &< V_{Tp}; \quad V_D - V_S < V_G - V_S - V_{Tp} \quad \text{(saturation)} \end{aligned}$$

where K_p, W_p, L_p, and V_{Tp} are the transconductance, channel width, channel length, and threshold voltage for the PMOS device respectively. Note that in SPICE, this model is selected by using the option *"LEVEL=1"* in the *.MODEL* statement.

It has been shown that (5.1) and (5.2) can be most easily solved, without much loss of accuracy, if we adopt an explicit formulation, where the voltages at the previous time step are used in solving for the drain currents [53].

We present an example to verify the method. Consider the circuit of a CMOS NAND shown in Fig. 5.1 where we have assumed an output capacitance of 1 pF and a small fictitious capacitance of 0.01 pF at the inner node. For simplicity, the MOS parameters are given as follows: $K_n = K_p = 10\mu A/V^2$, $W_n = W_p = L_n = L_p = 5\mu m$, $V_{Tn} = -V_{Tp} = 0.75V$ and $Vdd = 6V$. Initial conditions are assumed to be available through the *.IC* statement. If needed, they can be computed using the method in [54]. Fig. 5.2 shows the simulation result of the circuit using both LIM and SPECTRE. Comparable accuracy is observed in both methods.

5.3 Multi-Rate Simulation for CMOS Circuit

It is well known that the choice of a stable time step for a LIM simulation depends on the capacitances at each node [35, 37]. Specifically, circuits that contain smaller capacitances require smaller time steps for a stable simulation. In Chapter 3, we have seen how the multi-rate technique has been applied to speed up LIM simulations without violating the stability criterion, whereby the circuit is first partitioned into smaller subcircuits and different time steps are used for different partitions depending on the maximum stable time step. In this work, we apply the multi-rate simulation technique on a node-by-node basis [55]. Instead of partitioning the entire circuit into smaller partitions, we evaluate each node with its own maximum stable time step depending on the value of the capacitance at that node. Since we are dealing with CMOS circuits, the problem is simplified as we are not dealing with branch inductances. We will illustrate this idea by means of an example.

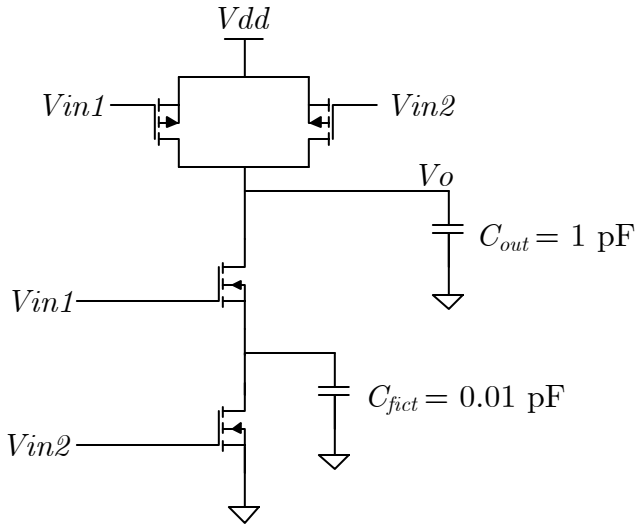


Figure 5.1: CMOS NAND.

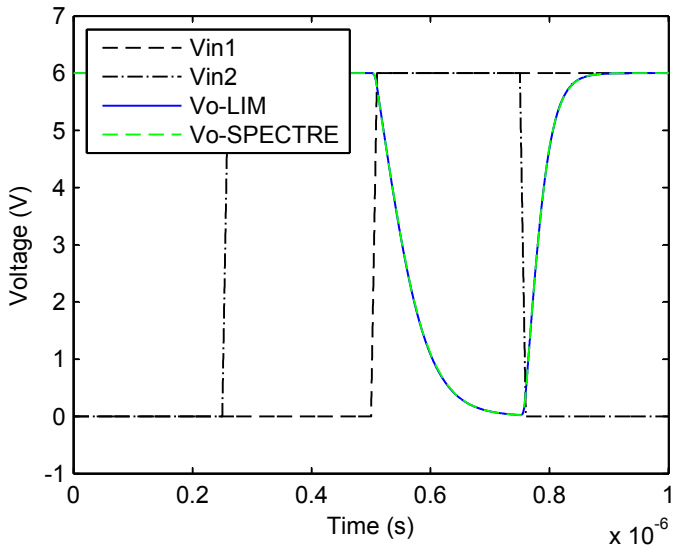


Figure 5.2: Simulation result of a CMOS NAND.

Consider again the CMOS NAND circuit shown in Fig. 5.1. The maximum stable time step for this circuit has been determined to be 0.1 ns. Note that this time step is due to the small fictitious capacitor, and if we split the circuit into two partitions as shown in Fig. 5.3, the upper node can be simulated using a time step of 10 ns without

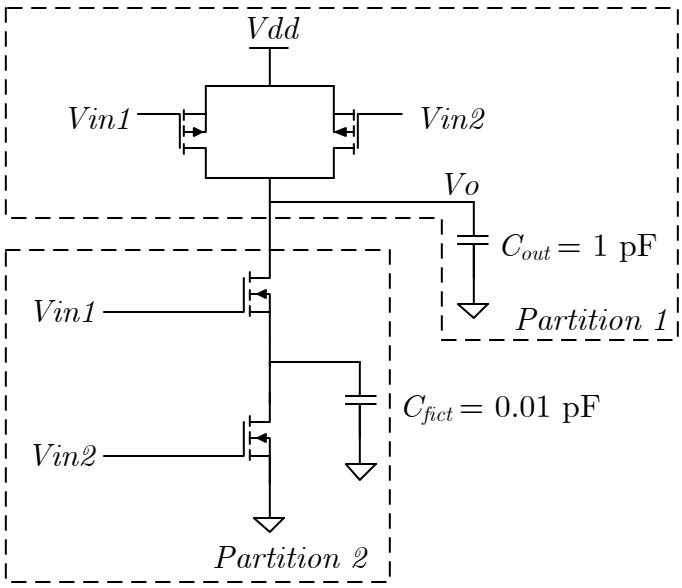


Figure 5.3: Partitioned CMOS NAND.

violating the stability criterion. Thus the idea is to simulate the circuit using the two time steps, one of each node. By doing so, we are able to speed up the simulation as the upper partition is only evaluated once for every 100 times of the lower partition. Fig. 5.4 shows the simulation of the circuit in Fig. 5.3 using the traditional LIM with time steps of 0.1 ns, 10 ns, and a multi-rate simulation with time steps of 0.1 ns and 10 ns. We see that the multi-rate simulation retains the accuracy of the 0.1 ns simulation while the simulation with time step of 10 ns results in an erroneous solution as expected.

5.4 Examples

In this section, two numerical examples will be presented. First a CMOS RAM circuit will be simulated in LIM and SPECTRE [22], a commercial circuit solver from Cadence Design Systems, in order to illustrate the speed improvement of LIM compared to SPICE based methods. Then a chain of ripple-carry adders will be

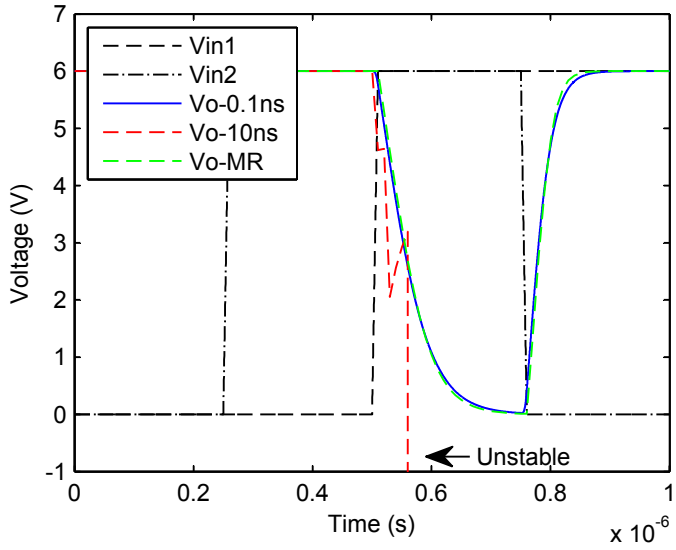


Figure 5.4: Simulation result of the partitioned CMOS NAND showing a LIM simulation with a time step of 0.1 ns (Vo-0.1ns), a LIM simulation with a time step of 10 ns (Vo-10ns) and a multi-rate LIM simulation (Vo-MR).

used to illustrate the application of the multi-rate simulation technique with CMOS devices.

5.4.1 RAM Circuit

In this example, a RAM circuit is simulated in LIM and in SPECTRE. The circuit contains 4850 nodes and 13,880 MOSFETs. A 1 pF capacitor is assumed to be present at each node to enable LIM. A time step of 0.5 ns is used in LIM in order to obtain a stable and accurate result while SPECTRE is allowed to determine its own suitable time step. The simulation length in both cases is 600 ns. Figs. 5.5 and 5.6 show the results at select nodes in both the LIM and SPECTRE simulation, respectively. We see that both methods produce comparable results. In terms of runtime, the LIM simulation requires 1.59 s for 1200 time steps while SPECTRE requires 22.24 s for 1146 time steps. Both simulations were performed on a Linux server with Intel Xeon 3.16 GHz processors and 32 GB of RAM. We see that LIM is about 14× faster in

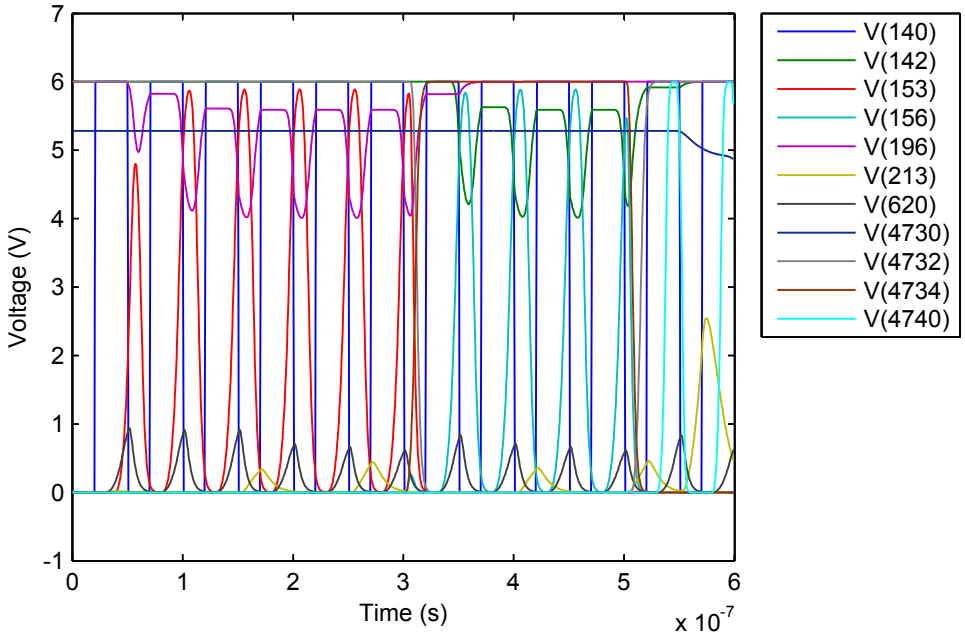


Figure 5.5: LIM simulation of RAM circuit.

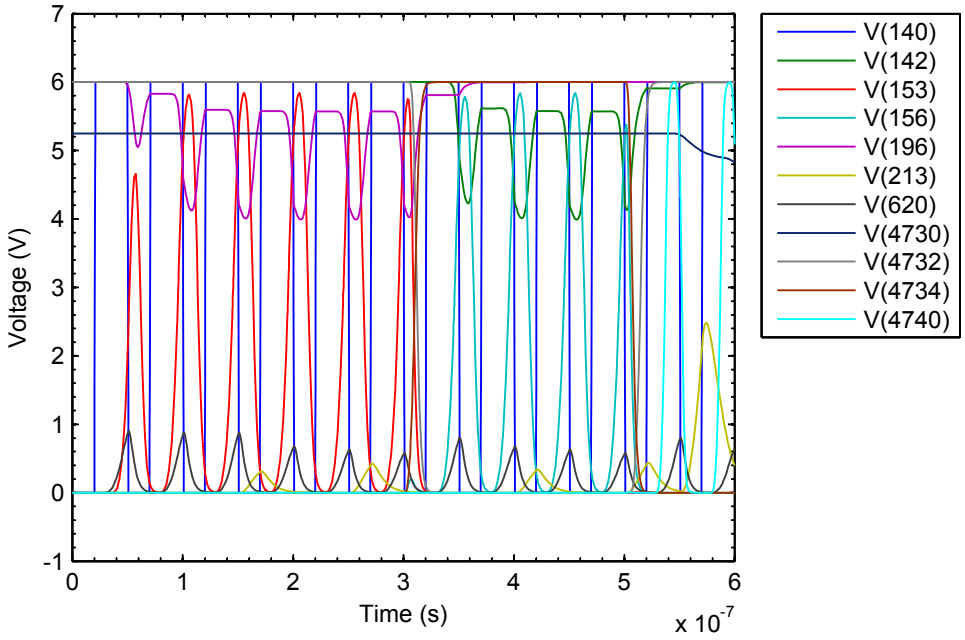


Figure 5.6: SPECTRE simulation of RAM circuit.

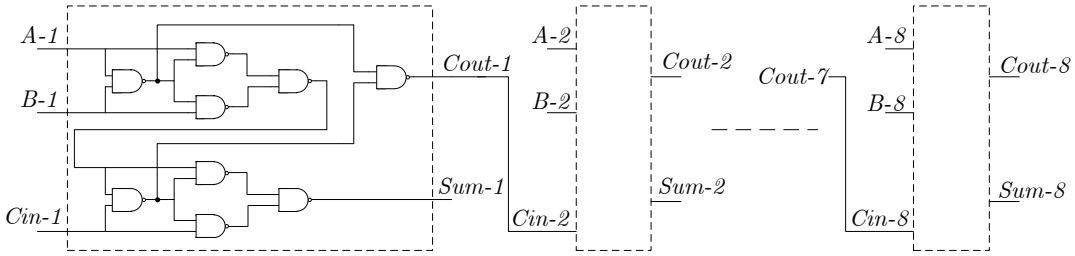


Figure 5.7: Chain of eight ripple-carry adders.

this example. The advantage of LIM in terms of runtime is expected to increase as the circuit size increases, as LIM exhibits a linear numerical complexity with respect to the number of nodes [36].

5.4.2 Ripple-Carry Adder

In this example, a chain of eight ripple-carry adders is simulated in order to illustrate an application of the multi-rate simulation technique. The circuit is shown in Fig. 5.7 where each NAND is as shown in Fig. 5.1. The regular LIM requires the use of a 0.1 ns time step in order to obtain a stable result, while the multi-rate LIM operates at time steps of 0.1 ns and 10 ns as explained in the previous section. The total simulation time is 2 μs, which results in 20,000 time steps. In the LIM simulation, the CMOS model is evaluated 5,759,712 times while the nodes are evaluated a total of 2,899,855 times. On the other hand, in the multi-rate LIM, the CMOS model is evaluated 2,908,800 times while the nodes are evaluated a total of 1,474,399 times. We see that by using the multi-rate technique, we are able to reduce the number of node and branch evaluations by almost a factor of two for this circuit. Fig. 5.8 shows the simulation result at the output of the first and last ripple-carry adders for both the regular LIM and the multi-rate LIM. Comparable accuracy is observed between the two.

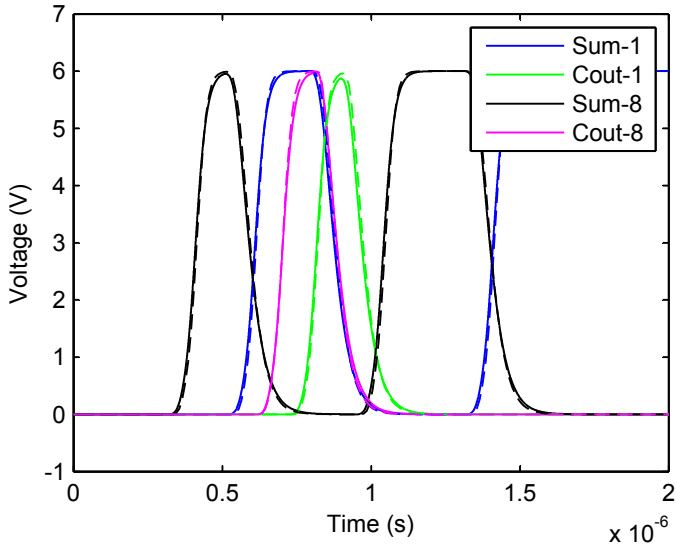


Figure 5.8: Simulation of ripple-carry adders in LIM (solid lines) and multi-rate LIM (dotted lines).

5.5 Summary

In this chapter, we have presented the formulation for the simulations of CMOS circuits in the LIM environment. Examples that illustrate the strength of the method in terms of speed and accuracy were presented. Finally, we note that the simulations of other nonlinear devices can be done in a similar fashion. For example, for the simulations of BJTs, large signal equations such as the Ebers-Moll equations can be used in place of (5.1) and (5.2).

CHAPTER 6

PLL SIMULATIONS

6.1 Introduction

This chapter presents an extension of the latency insertion method (LIM) to the simulations of analog devices, particularly to phase-locked loops (PLLs). PLLs are extensively used in modern wireless communication and high-speed devices. They can be employed to perform an array of functions, ranging from frequency synthesizers to clock recovery and data synchronizers. However, despite their prominence in applications, simulations of PLLs still constitute a significant challenge to the industry today. Traditional simulations of PLLs at the transistor level, albeit accurate, are often prohibitively slow due to the dual time scale problem. The high frequency of the embedded voltage-controlled oscillator necessitates the use of very small simulation time steps, while the overall loop bandwidth is typically orders of magnitude lower which results in very long simulation time in order to observe the dynamic behavior of the system. As a result, some designers resort to analytical expression based and behavioral macromodeling simulations of PLLs [56, 57]. While these methods offer significant speed-ups compared to a full transistor level simulation, an overly simplified linear model can often neglect key nonlinear behaviors, resulting in erroneous response of the PLL. In addition, complex behavioral models can be cumbersome to implement and might not be easily integrated into a system level simulation.

In this chapter, we will examine the usage of LIM for the simulations of PLLs. First, a behavioral level simulation will be performed using the PLL governing equations.

By exploiting the latency in the formulation, along with a leapfrog time-stepping discretization scheme, we solve the PLL governing equations without the formulation of complex, high-order differential equations. In addition, nonlinearities of the PLL components can be easily integrated into the existing formulation and extensions to higher-order PLLs are straightforward. Second, we show an example of simulating a PLL at the transistor level using LIM. This will illustrate the capabilities of LIM to perform a transistor level simulation of analog devices when higher accuracies are desired.

6.2 Behavioral Simulations of PLLs Based on a Leapfrog Voltage-Phase Formulation

In this section, we present a novel and simple behavioral model based simulation method for PLLs. The method exploits the latency in the PLL formulation and utilizes a leapfrog time-stepping discretization scheme to solve for the transient response of the PLL. Various PLL dynamic responses such as lock-in, pull-in and pull-out conditions are simulated and comparisons with analytical solutions are depicted when available. In addition, the method is shown to be able to capture nonlinear behaviors of the PLL. Due to the formulation in the voltage-phase domain, the method does not suffer from the dual time scale problem which is a main issue in full transistor level simulations of PLLs.

Fig. 6.1 shows a block diagram of a PLL, consisting of a phase detector (PD), a low-pass loop filter (LPF) and a voltage-controlled oscillator (VCO). For simplicity, the frequency divider, which is often used in a synthesizer, is assumed to be unity. In order to overcome the dual time scale problem, we will adopt a phase-domain characterization of the PD and the VCO. In Fig. 6.1, the PD typically governs the main nonlinear behavior of the PLL due to its inherent nonlinearity. For example,

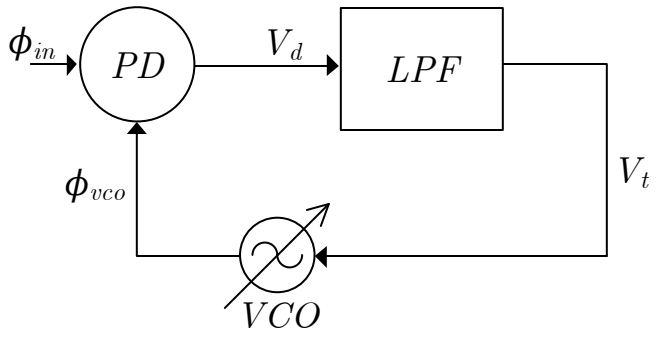


Figure 6.1: Block diagram of a PLL.

when an analog multiplier is used as a PD, its output signal is given by

$$V_d(t) = K_D \sin\left(\phi_e(t)\right) \tag{6.1}$$

where $\phi_e(t)$ is the phase error defined as

$$\phi_e(t) = \phi_{in}(t) - \phi_{vco}(t). \tag{6.2}$$

Note that the sum output term has been neglected since it will be filtered out by the LPF. Alternative implementations of the PD exist, for example by using a JK flip-flop in a digital PD. In that case, (6.1) can be replaced by a sawtooth function [58].

The LPF in Fig. 6.1 is modeled by its transfer function. For example, for an active second order filter, we obtain [58]

$$\frac{V_t(s)}{V_d(s)} = \frac{1 + \tau_2 s}{\tau_1 s}. \tag{6.3}$$

Rearranging the terms in (6.3) and taking the inverse Laplace transform we obtain

$$\tau_1 \frac{d}{dt} V_t(t) = V_d(t) + \tau_2 \frac{d}{dt} V_d(t). \tag{6.4}$$

For higher order filters, (6.4) can be modified accordingly.

Finally, the VCO is modeled by

$$\frac{d}{dt}\phi_{vco}(t) = K_V V_t(t) + \omega_{offset} \tag{6.5}$$

where the output frequency has been substituted as the derivative of the phase and ω_{offset} is the free running frequency of the VCO. Substituting (6.2) into (6.5) yields

$$\frac{d}{dt}\left(\phi_{in}(t) - \phi_e(t)\right) = K_V V_t(t) + \omega_{offset} \tag{6.6}$$

$$\frac{d}{dt}\phi_e(t) = \omega_{in} - \omega_{offset} - K_V V_t(t) \tag{6.7}$$

Next, substituting (6.1) into (6.4) and rearranging the terms we obtain

$$\frac{d}{dt}V_t(t) = \frac{K_D}{\tau_1}\sin\phi_e(t) + \frac{\tau_2}{\tau_1}\frac{d}{dt}\left(K_D \sin\phi_e(t)\right). \tag{6.8}$$

In order to solve (6.7) and (6.8), we apply a leapfrog discretization scheme where $V_t(t)$ and $\phi_e(t)$ are collated in half time steps to generate sequences of the form $V_t^{n-1/2}$, $V_t^{n+1/2}$, $V_t^{n+3/2}$ for the tune voltages and ϕ_e^n, ϕ_e^{n+1}, ϕ_e^{n+2} for the phase errors. This is similar to LIMs solution of the Kirchhoff's voltage and current law circuit equations. Applying this to (6.7) and (6.8) we obtain

$$\phi_e^{n+1/2} = \phi_e^{n-1/2} + \Delta t\left(\omega_{in} - \omega_{offset} - K_V V_t^n\right) \tag{6.9}$$

$$V_t^{n+1} = V_t^n + \frac{\Delta t}{\tau_1}\left(K_D \sin\phi_e^{n+1/2}\left(1 + \frac{\tau_2}{\Delta t}\right) - \frac{\tau_2}{\Delta t}K_D \sin\phi_e^{n-1/2}\right) \tag{6.10}$$

The transient solution of the PLL can then be calculated by alternating the computations of (6.9) and (6.10) as time progresses. Note that this method avoids the formulation of a complex high-order differential equation. In addition, nonlinearities

Table 6.1: PLL parameters.

K_D	K_V	τ_1	τ_2
$5/(2\pi)$	$2\pi(3 \times 10^5)$	4.385×10^{-6}	1.592×10^{-6}

in the PLL components can be readily integrated into the modeling equations of (6.1) and (6.5).

Next, we apply the developed method to simulate the lock-in and pull-in or acquisition process of a PLL. An example PLL is used where the parameters are shown in Table 6.1.

First, the PLL is assumed to be in a locked condition and a small unit step change is applied to the input frequency. The dynamics of the PLL as it relocks is monitored and the output frequency and phase error are plotted in Fig. 6.2 and Fig. 6.3 respectively. For this small perturbation, the phase error is sufficiently small and the PLL operates in the linear region where

$$\sin \phi_e(t) \approx \phi_e(t) \tag{6.11}$$

Using this approximation in the linear region, an analytical solution of the PLL can be calculated by taking the inverse Fourier transform of the closed-loop frequency response multiplied by the unit step function. This method is presented in detail in [59]. The output frequency and phase error calculated using the analytical solution are superimposed on Fig. 6.2 and Fig. 6.3 respectively. We see a good agreement between the two methods.

Next, a larger step change of 500 kHz is applied to the input frequency. This simulates the acquisition process which typically occurs when the PLL is first powered up or when subjected to a large perturbation. In this case, the PLL leaves the linear region, and exhibits a highly nonlinear behavior. The output frequency is simulated

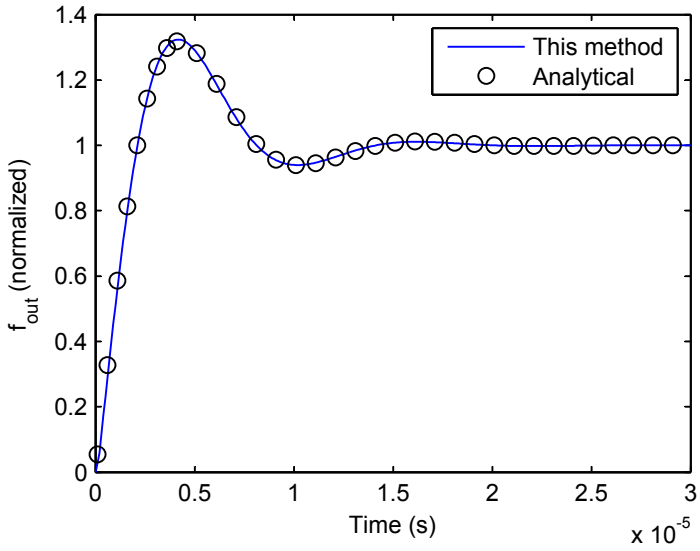


Figure 6.2: Output frequency of PLL during lock-in.

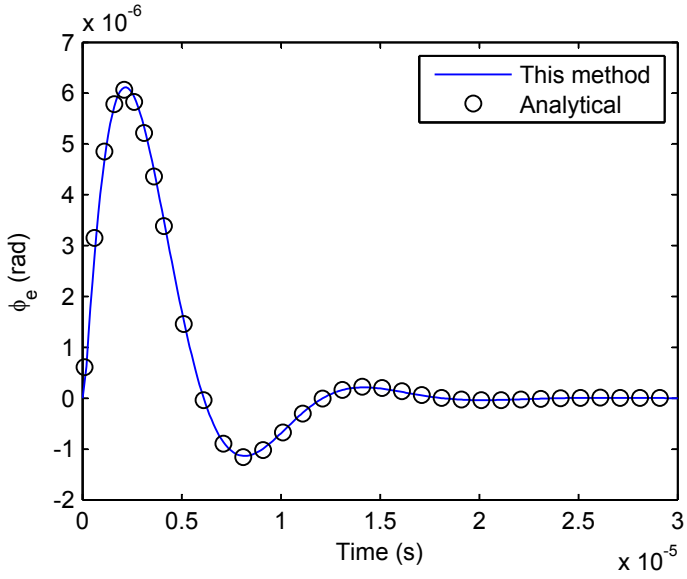


Figure 6.3: Phase error of PLL during lock-in.

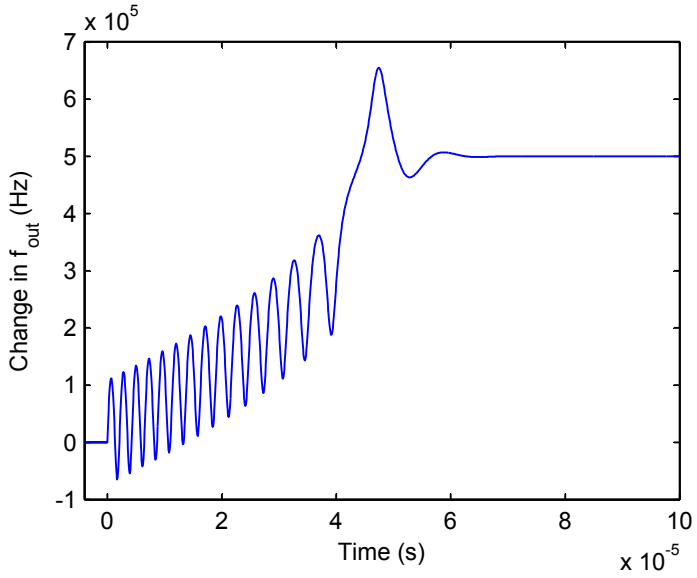


Figure 6.4: Output frequency of PLL during acquisition.

and shown in Fig. 6.4. Note that for this nonlinear process, simple analytical solutions which assume the linearity of the PLL are no longer valid.

Finally, the PLL is subjected to an even larger step change of 2 MHz in input frequency. The output frequency is simulated and plotted in Fig. 6.5. In this case, the PLL struggles to acquire lock and no indication of locking can be seen in the time frame simulated.

Before concluding this section, we note that all the simulations depicted using the method took less than one second to run on an AMD 3 GHz desktop computer with 4 GB of RAM.

6.3 Transistor Level Simulations of PLLs Using LIM

In this section, a transistor level simulation of a PLL will be performed using LIM. The PLL is represented as in Fig. 6.1. An XOR gate is used as a phase detector and a circuit diagram of it is given in Fig. 6.6. The loop filter is a first order low-pass filter as shown in Fig. 6.7 and the VCO is shown in Fig. 6.8, where an inverter is used

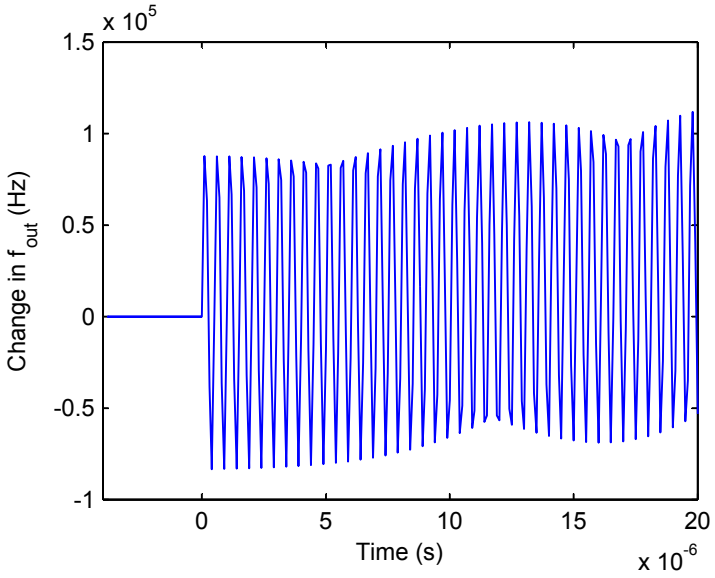


Figure 6.5: Output frequency of PLL for a large step change in input frequency illustrating a pull-out process.

to convert between the generated sine wave and a square wave. For simplicity, all the MOS transistors are assumed to have parameters as follows: $K_n = K_p = 20\,\mu\text{A/V}^2$, $W_n = W_p = 10\ \mu\text{m}$, $L_n = L_p = 1\ \mu\text{m}$, $V_{Tn} = -V_{Tp} = 0.75\,\text{V}$ and $Vdd = 5\,\text{V}$. The variable capacitors in the VCO have values of 195.8 pF − 10 pF/V_t which gives a free running frequency of 36 MHz ($V_t = 0$) and a tuning characteristic of 1 MHz/V_t. No particular implementation is assumed for the varactors in this simulation. In practice, they are often implemented as reverse biased diodes, PMOS transistors with the drain, source and bulk connected together, or an array of these to achieve wider tuning ranges [60]. Finally, we note that since an XOR gate is used as a phase detector, the PLL is designed to have a center frequency of 38.5 MHz which corresponds to a tuning voltage of $Vdd/2 = 2.5\,\text{V}$.

The PLL is then subjected to a square wave input signal V_{in}, with rise and fall times of 1 ns, a pulse width of 11.987 ns and a period of 25.974 ns which corresponds to a frequency of 38.5 MHz. The magnitude of the pulse is 5 V. V_{BIAS} is 1 V and $I_{kickstart}$ is a current pulse with rise and fall times of 1 ns, a pulse width of 3 ns and a magnitude

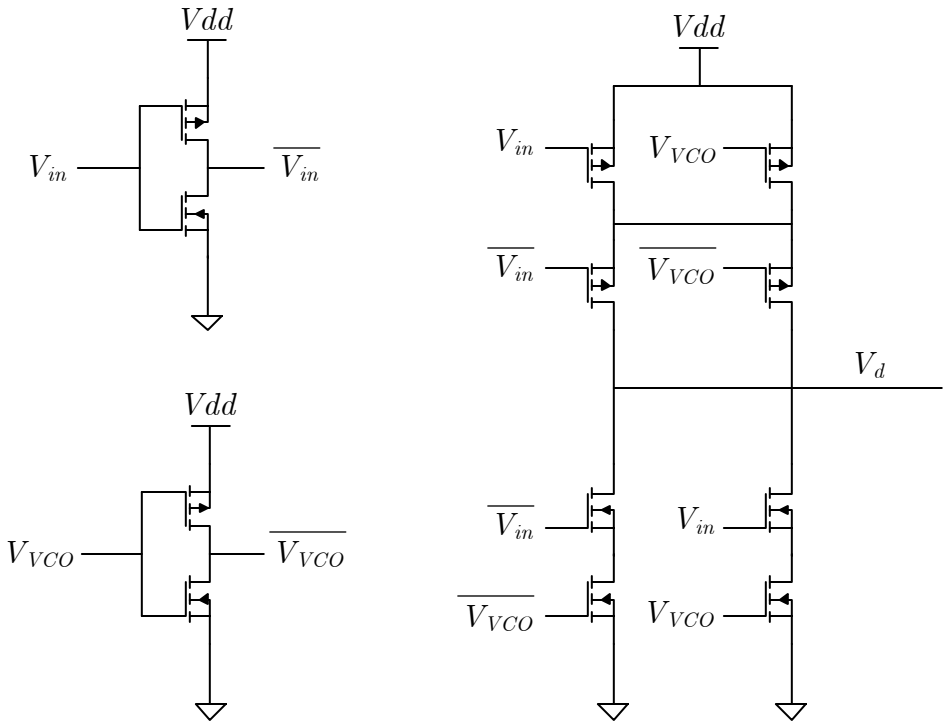


Figure 6.6: XOR phase detector.

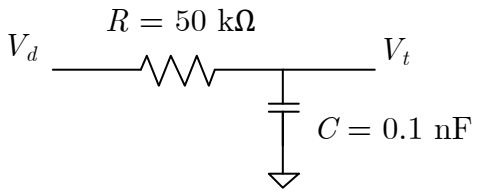


Figure 6.7: Low pass filter.

of 0.5 A which is used solely for the purpose of simulation. All the node voltages in the circuit are assumed to begin at zero and the behavior of the PLL as it locks to the input signal is simulated using both LIM and SPECTRE [22], a commercial circuit solver from Cadence Design Systems. This corresponds to an acquisition process when the PLL is first powered up. In LIM, small fictitious inductors and capacitors of magnitude 1 nH and 0.01 pF respectively are inserted into branches and nodes without latencies in order to enable the method. The tuning voltage V_t is plotted

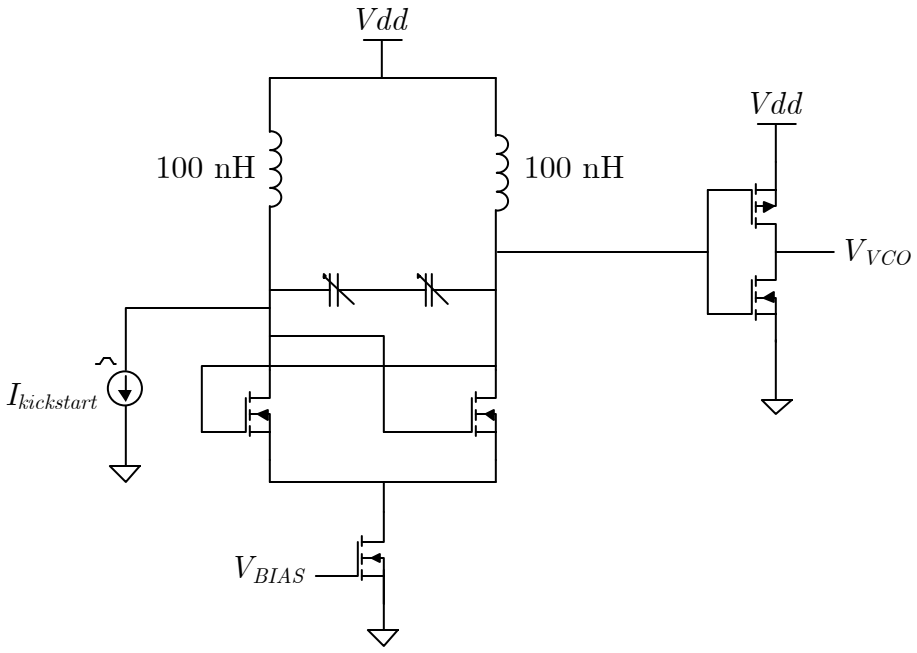


Figure 6.8: VCO.

in Fig. 6.9 for a 75 μs simulation time for both LIM and SPECTRE. We see a very good agreement between the two methods. In terms of runtime, LIM required a time step of 5 ps for a stable simulation which resulted in a runtime of 10.82 s for 15,000,000 time steps. SPECTRE was able to obtain an accurate result with a time step of 0.1 ns which resulted in a runtime of 34.78 s for 989,514 time steps. (Note that SPECTRE automatically adjusts the time step for convergence of the embedded Newton-Raphson iteration.) This illustrates a typical scenario for a PLL simulation at the transistor level, where a large number of simulation time steps is required due to the dual time scale problem, where the high frequency of the embedded voltage-controlled oscillator necessitates the use of a very small simulation time step, while the lower overall loop bandwidth determines the total simulation time that has to be performed to observe the dynamic behavior of the system. In both cases, the simulations are performed on a Linux server with Intel Xeon 3.16 GHz processors and 32 GB of RAM. We see that the runtime for both methods are comparable for

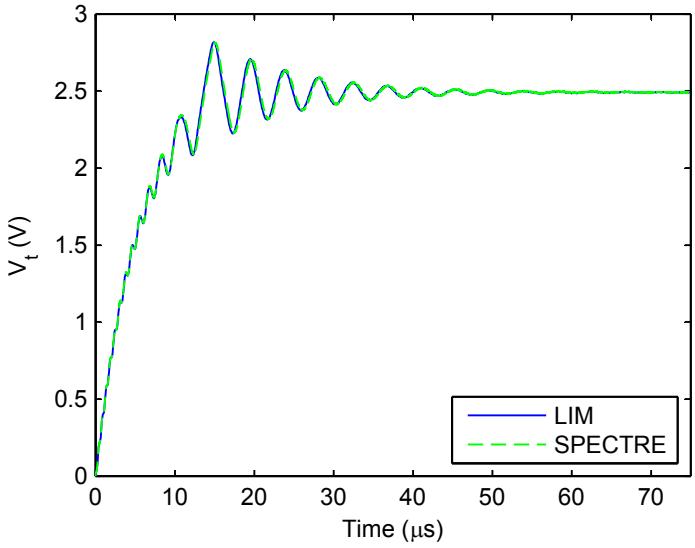


Figure 6.9: PLL tuning voltage during acquisition.

this small example. However, we expect that for larger circuits, such as when a PLL is embedded into a larger system, LIM would outperform SPECTRE as we have seen in Chapter 2 that LIM exhibits a linear numerical complexity with respect to the number of nodes.

Next, the same PLL is simulated using the behavioral model approach presented in Section 6.2. Since an XOR gate is used as a phase detector, (6.1) is replaced by a triangular function from 0 to 5 with period π:

$$V_d(t) = 2.5 \cdot \frac{2}{\pi}\left((\phi_e(t)+\pi/2) - \pi\left\lfloor \frac{(\phi_e(t)+\pi/2)}{\pi} + \frac{1}{2}\right\rfloor\right)(-1)^{\left\lfloor \frac{(\phi_e(t)+\pi/2)}{\pi} - \frac{1}{2}\right\rfloor} + 2.5 \tag{6.12}$$

where $\lfloor x \rfloor$ represents the floor function of x. In addition, (6.4) is replaced by a passive first order filter:

$$V_t(t) + \tau_1 \frac{d}{dt} V_t(t) = V_d(t) \tag{6.13}$$

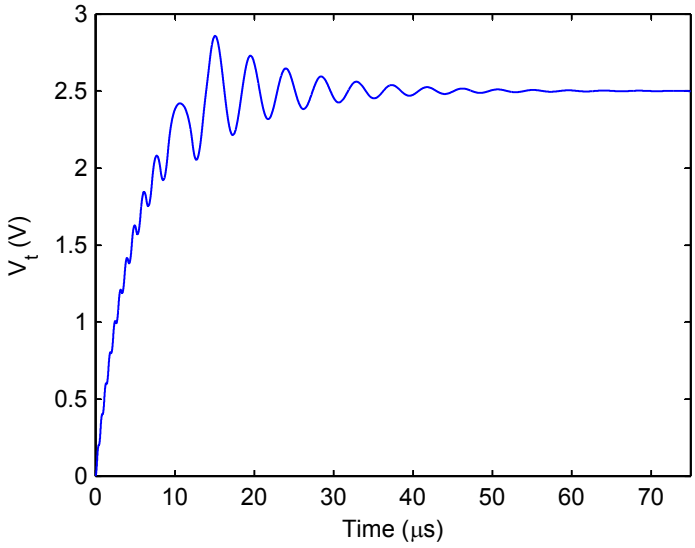


Figure 6.10: PLL tuning voltage during acquisition from behavioral model.

where $\tau_1 = RC$. The VCO is modeled as in (6.5) with an offset frequency of 38.5 MHz corresponding to a tuning voltage of 2.5 V:

$$\frac{d}{dt}\phi_{vco}(t) = (2\pi \cdot 38.5 \times 10^6) - K_V \left(V_t(t) - 2.5\right). \tag{6.14}$$

The remaining parameters of the PLL are $K_D = 1$, since the phase detector gain has been included in (6.12), and $K_V = 2\pi(10^6)$ corresponding to a 1 MHz/V tuning characteristic of the VCO. Fig. 6.10 shows the tuning voltage V_t from the behavioral model simulation. We see a good agreement with the transistor level simulation shown in Fig. 6.9. Some slight differences between the outputs of the two methods are expected to be caused by the nonideal behavior of the actual circuit that is not captured in the behavioral modeling. This will be investigated next. Before we proceed, we remark that the behavioral model simulation took less than one second of runtime on the same computer.

In order to examine the accuracy of the modeling equations used in the behavioral level simulation, we plot the average output voltage of the phase detector shown in

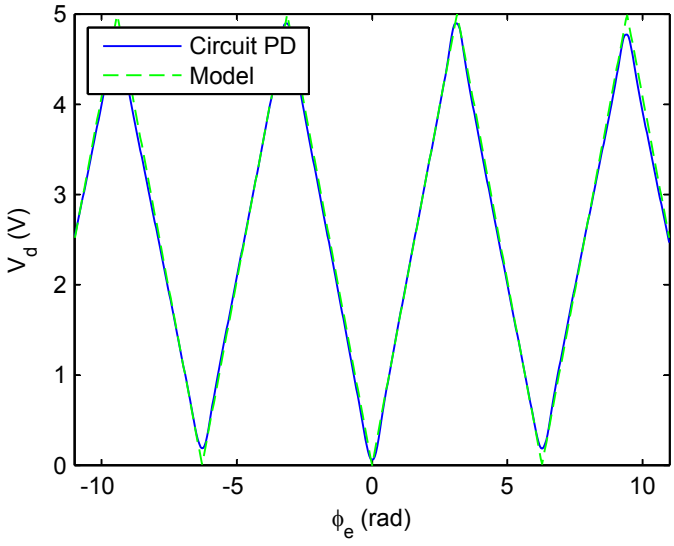


Figure 6.11: Response of XOR phase detector.

Fig. 6.6 as the phase error between the two inputs is varied from -3.5π to 3.5π, along with the modeling equation in (6.12). This is shown in Fig. 6.11. In addition, the output frequency of the VCO shown in Fig. 6.8 is plotted as the tuning voltage, V_t is varied from 0 V to 5 V, along with the linear approximation used in (6.14). This is shown in Fig. 6.12. Two notable differences between the responses of the actual circuit and the model are: (1) for very small ($\phi_e \approx 0,\ 2\pi,\ ...$) and very large ($\phi_e \approx \pi,\ 3\pi,\ ...$) phase errors, the response for the actual circuit only approaches the ideal response of 0 V and 5 V respectively, due to the delay of the internal components of the PD, and (2) the tuning characteristic of the actual VCO deviates slightly from the ideal approximation used in the model. This would explain the slight discrepancies between the results in Fig. 6.9 and Fig. 6.10. If needed, the modeling equations in (6.12) and (6.14) can be tuned to better capture the exact behaviors of the PD and the VCO. This would, for example, be useful in a bottom-up design approach where the individual component parameters are first extracted and then used in the design and simulation of the final overall system.

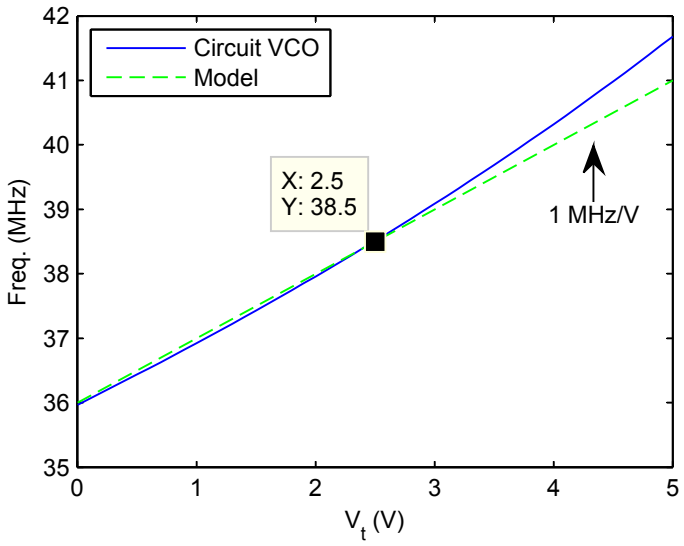


Figure 6.12: Response of VCO.

6.4 Additional Simulations and Discussions

In this section, we present some additional simulations of the PLL and include some discussions of the results. First, the same PLL used in the previous section is subjected to a square wave input signal V_{in}, with rise and fall times of 1 ns, a pulse width of 11.8205 ns and a period of 25.641 ns which corresponds to a frequency of 39 MHz. The magnitude of the pulse is 5 V. The simulation results from both LIM and SPECTRE are shown in Fig. 6.13. In this case, we see that the input frequency is outside the PLL pull-in range and the PLL does not acquire lock. The same simulation is also performed using the behavioral model approach and is shown in Fig. 6.14. Comparable accuracy is observed between all three methods.

Next, a long simulation is performed on the PLL. This is shown in Fig. 6.15. First the PLL is subjected to a square wave input signal V_{in}, with rise and fall times of 1 ns, a pulse width of 11.987 ns and a period of 25.974 ns which corresponds to a frequency of 38.5 MHz. Once the PLL has acquired lock the input signal is changed twice, first to 38.3 MHz at 75 μs and then again to 38.6 MHz at 130 μs. We see that the PLL

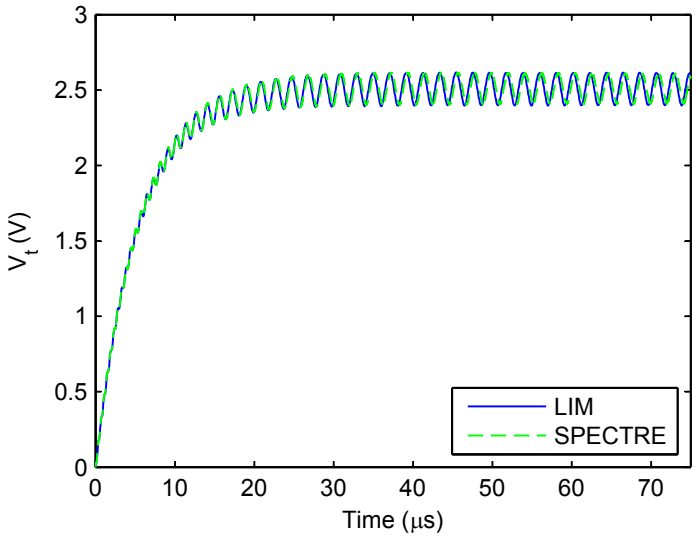


Figure 6.13: PLL tuning voltage for a 39 MHz input signal.

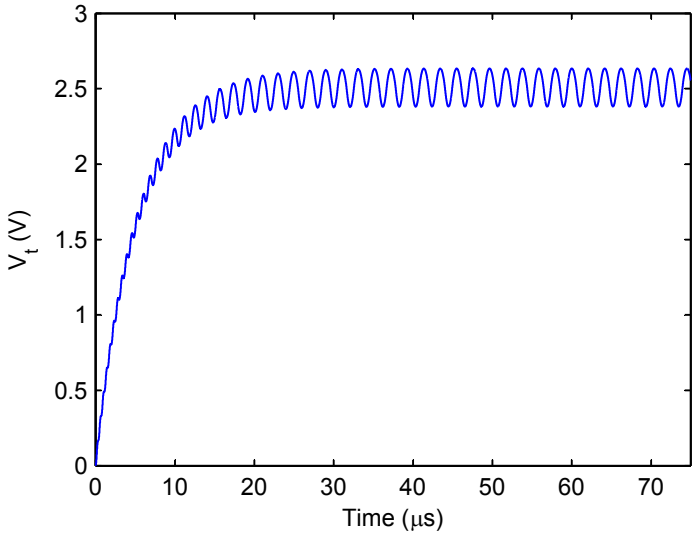


Figure 6.14: PLL tuning voltage for a 39 MHz input signal from behavioral model.

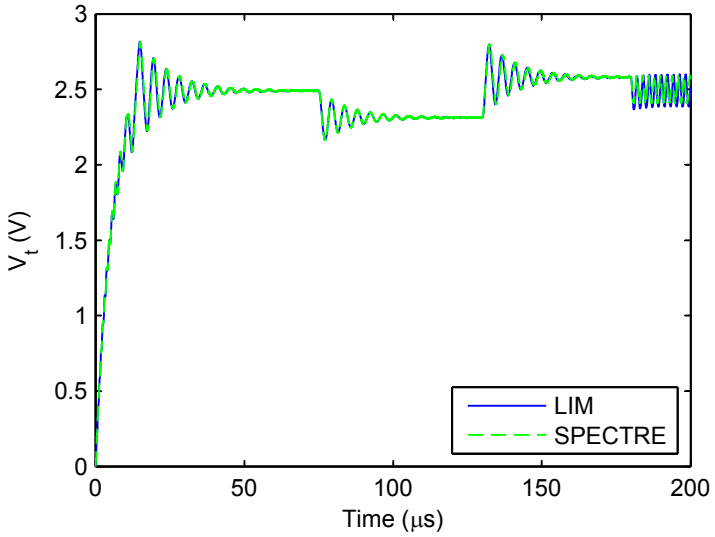


Figure 6.15: PLL tuning voltage for a long simulation.

is able to track the input signal and maintain a locking condition. Finally, at 180 μs, the input signal is changed to 38 MHz. In this case, the change is large enough that the PLL loses lock. The same simulation is also performed using the behavioral model approach and is shown in Fig. 6.16. Comparable accuracy is observed between all three methods.

In all the simulations in this section, the runtimes for both LIM and SPECTRE are comparable to those recorded in the previous section. The runtime for the LIM simulation can be improved by using larger fictitious latency elements which would allow the use of a larger time step without violating the stability criterion. Doing so, however, would result in some loss of accuracy. For instance, consider the example simulated in Fig. 6.9. If the fictitious capacitors were increased to 0.1 pF, the time step could be increased to 10 ps which reduces the runtime to 5.51 s. If the fictitious inductors were also increased to 10 nH, the time step could be further increased to 50 ps which further reduces the runtime to 1.28 s. The outputs from these simulations are shown in Fig. 6.17. We see a clear tradeoff between speed and accuracy. In

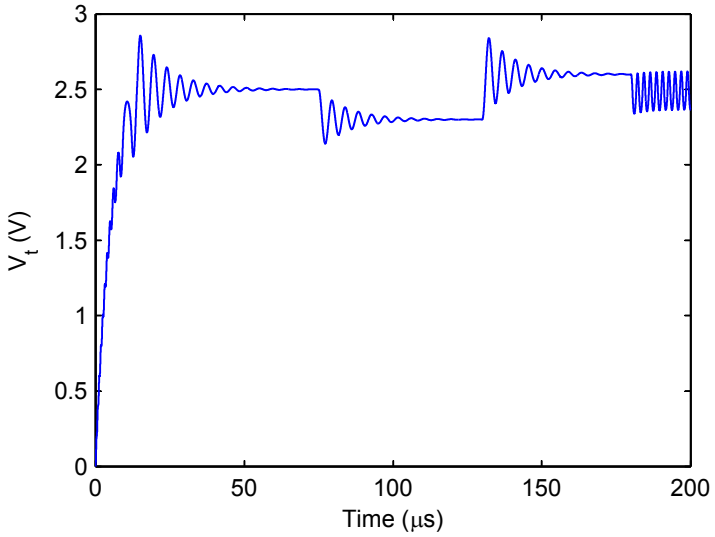


Figure 6.16: PLL tuning voltage for a long simulation from behavioral model.

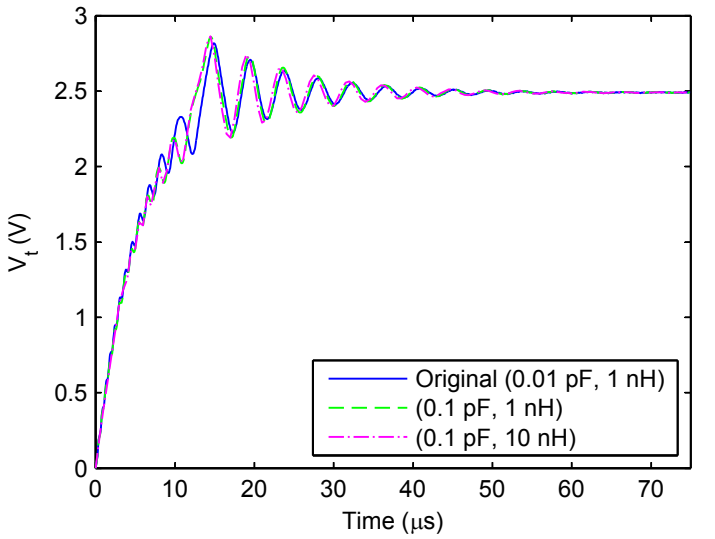


Figure 6.17: PLL tuning voltage for different fictitious latency values.

addition, this also suggests that the insertion of fictitious latencies can be utilized as a way to perform dynamic time step control in LIM, which could be the subject of a future research.

6.5 Summary

In this chapter, we have presented two methods for the simulations of PLLs based on the latency insertion method. First, the behavioral model approach is depicted as a fast, simple and efficient methodology for the simulations of PLLs. The modularity of the method allows the incorporations of nonlinear effects in the PLL components in a straightforward manner. The resulting equations are solved for in a leapfrog time-stepping scheme by taking advantage of the latency in the formulation. It is shown that the method is able to capture intrinsic behaviors of PLLs and exhibits good correlations with transistor level simulations when accurate models and design parameters are available. Second, a transistor level simulation of a PLL is performed using LIM and comparisons with an existing commercial simulator is shown. This illustrates the capabilities of LIM as an analog circuit simulation tool. Based on the findings in the previous chapters, LIM is expected to be most beneficial when the size of the circuit is large.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this work, we have presented a fast, multi-purpose circuit simulator using the latency insertion method. Advancements in LIM, such as the ability to simulate circuits with dependent sources using the Block-LIM formulation, were depicted. A detailed stability analysis of the method was also performed which led to the formulation of the multi-rate or partitioned latency insertion method (PLIM) where circuits with partitions of multiple latencies could be simulated using different time steps for each partition.

Next, a detailed formulation of blackbox macromodeling in the LIM environment was carried out. Circuits characterized by their terminal responses were modeled either by a passive MOR technique utilizing the vector fitting method or by a fast convolution approach. Various aspects of each method such as stability, passivity and causality were also addressed. This concluded with a comparative study of the two methods and its incorporation into a LIM simulation.

The simulations of nonlinear devices, such as CMOS circuits, were also presented where the accuracy of the method was verified and the speed improvement depicted in comparison to traditional SPICE-based methods. The multi-rate simulation technique was also extended to CMOS circuits.

Finally, the subject of analog circuit simulations was explored by utilizing LIM for the simulation of phase-locked loops. Two approaches for the simulations of PLLs

were presented. First a behavioral model simulation was described using the PLL governing equations. Simulation examples were shown that illustrated the method's strength in terms of speed and accuracy, when used in either a top-down or bottom-up design approach. Second, a full transistor level simulation of an example PLL was performed using LIM and comparisons with existing commercial simulators are depicted. We conclude that LIM is well suited for the simulations of analog circuits and shows great potential moving forward.

7.2 Future Work

While some significant contributions have been presented in this dissertation towards applying the latency insertion method as a fast, multi-purpose circuit simulator, there are undoubtedly more challenges and further improvements that can be made to the method. We summarize some prospective future work on the subject here.

The first, and perhaps most prominent, aspect of the method that can be better investigated and explored is the issue of latency insertion. As mentioned in Chapter 2, LIM requires the presence of latencies in the circuit to perform the leapfrog time stepping algorithm. When they are not present, small fictitious values are inserted in order to enable the method. This leads to a clear tradeoff between speed and accuracy. Smaller fictitious values increase the accuracy of the simulation but they also decrease the maximum stable time step, which results in longer simulation times. The study and development of a fully automated process of latency insertion, for example based on user specified threshold error values, would be a significant contribution to the LIM method and would be very valuable towards marketing LIM as a robust computer-aided design tool for the engineering community.

A second, interesting topic that can be explored, is on the parallelization of LIM for applications on multi-core CPUs and GPUs. As the semiconductor industry ap-

proaches the limit of Moore's law, microprocessor designers have shifted from the former trend of increasing the clock frequency for faster computation, to the present trend of stacking multiple cores on a single processor. This has led to the need of multi-threaded programs which take advantage of the availabilities of these additional processing cores for enhanced performance. LIM too could benefit greatly from a parallel implementation. We believe that the distributed nature of the partitioned latency insertion method (PLIM), presented in Chapter 3, would make it a suitable candidate for parallelization and provides a good starting point for future research on the subject.

Another possible future direction that can be taken on the subject is on the incorporation of field solvers into a LIM simulation. Such a hybrid field-circuit solver would be beneficial as it would be able to take advantage of the high accuracy of field solvers for the solutions of microwave components, while at the same time allow fast and simple solutions of lumped linear and nonlinear components using circuit simulation techniques. Due to its resemblance to the finite-difference time-domain (FDTD) method, a natural initial choice would be on the formulation of an FDTD-LIM hybrid. While the idea of a synergy between field and circuit solvers is not new, we feel that earlier works have only scraped the surface of the potential that could be unlocked with a robust and comprehensive implementation. A short list of related past work on the subject that would serve as a good starting point for future research on this topic would include [61–65].

A fourth and final future work proffered here is on the formulation of a LIM-SPICE hybrid, where some initial work can be found in the literature [66,67]. Despite its inherent shortcoming in solving large circuits, SPICE still benefits from a wide array of features such as the abundance of device modeling in its environment and its popularity in the industry. On the other hand, LIM is most adept at solving large, high-frequency circuits. Thus, an integration of LIM as a functional block

into SPICE would allow for the fast simulations of large interconnect networks using LIM while at the same time offering compatibility with any other components and terminations that are supported by SPICE. Furthermore, an association with SPICE could potentially raise more awareness of LIM which is invaluable towards marketing LIM to the engineering community as a whole.

REFERENCES

[1] R. Achar and M. Nakhla, "Simulation of high-speed interconnects," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 693–728, May 2001.

[2] A. Ruehli and A. Cangellaris, "Progress in the methodologies for the electrical modeling of interconnects and electronic packages," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 740–771, May 2001.

[3] F. Branin, "Transient analysis of lossless transmission lines," *Proceedings of the IEEE*, vol. 55, no. 11, pp. 2012–2013, Nov. 1967.

[4] F. Y. Chang, "Transient analysis of lossless coupled transmission lines in a nonhomogeneous dielectric medium," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 18, no. 9, pp. 616–626, Sep. 1970.

[5] F. Y. Chang, "The generalized method of characteristics for waveform relaxation analysis of lossy coupled transmission lines," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 37, no. 12, pp. 2028–2038, Dec. 1989.

[6] C. Paul, *Analysis of Multiconductor Transmission Lines*. New York, NY: Wiley, 1994.

[7] W. Beyene and J. Schutt-Aine, "Accurate frequency-domain modeling and efficient circuit simulation of high-speed packaging interconnects," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 45, no. 10, pp. 1941–1947, Oct. 1997.

[8] B. Gustavsen and A. Semlyen, "Rational approximation of frequency domain responses by vector fitting," *Power Delivery, IEEE Transactions on*, vol. 14, no. 3, pp. 1052–1061, Jul. 1999.

[9] S. Grivet-Talocia, "Package macromodeling via time-domain vector fitting," *Microwave and Wireless Components Letters, IEEE*, vol. 13, no. 11, pp. 472–474, Nov. 2003.

[10] D. Deschrijver, B. Haegeman, and T. Dhaene, "Orthonormal vector fitting: A robust macromodeling tool for rational approximation of frequency domain responses," *Advanced Packaging, IEEE Transactions on*, vol. 30, no. 2, pp. 216–225, May. 2007.

[11] D. Deschrijver, M. Mrozowski, T. Dhaene, and D. De Zutter, "Macromodeling of multiport systems using a fast implementation of the vector fitting method," *Microwave and Wireless Components Letters, IEEE*, vol. 18, no. 6, pp. 383–385, Jun. 2008.

[12] D. Saraswat, R. Achar, and M. Nakhla, "A fast algorithm and practical considerations for passive macromodeling of measured/simulated data," *Advanced Packaging, IEEE Transactions on*, vol. 27, no. 1, pp. 57–70, Feb. 2004.

[13] S. Grivet-Talocia, "Passivity enforcement via perturbation of Hamiltonian matrices," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 9, pp. 1755–1769, Sep. 2004.

[14] D. Saraswat, R. Achar, and M. Nakhla, "Global passivity enforcement algorithm for macromodels of interconnect subnetworks characterized by tabulated data," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 13, no. 7, pp. 819–832, Jul. 2005.

[15] A. Lamecki and M. Mrozowski, "Equivalent SPICE circuits with guaranteed passivity from nonpassive models," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 55, no. 3, pp. 526–532, Mar. 2007.

[16] B. Gustavsen and A. Semlyen, "Fast passivity assessment for S-parameter rational models via a half-size test matrix," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 56, no. 12, pp. 2701–2708, Dec. 2008.

[17] E. Chiprout and M. Nakhla, *Asymptotic Waveform Evaluation and Moment Matching for Interconnect Analysis*. Boston, MA: Kluwer, 1993.

[18] D. Mardare and J. LoVetri, "The finite-difference time-domain solution of lossy MTL networks with nonlinear junctions," *Electromagnetic Compatibility, IEEE Transactions on*, vol. 37, no. 2, pp. 252–259, May. 1995.

[19] A. Orlandi and C. Paul, "FDTD analysis of lossy, multiconductor transmission lines terminated in arbitrary loads," *Electromagnetic Compatibility, IEEE Transactions on*, vol. 38, no. 3, pp. 388–399, Aug. 1996.

[20] J. Schutt-Aine, "Latency insertion method (LIM) for the fast transient simulation of large networks," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 48, no. 1, pp. 81–89, Jan. 2001.

[21] L. W. Nagel, "SPICE2, A computer program to simulate semiconductor circuits," Univ. California, Berkeley, Tech. Rep. ERL-M520, 1975.

[22] *Virtuoso Advanced Analysis Tools User Guide, Cadence 5.1.41 Spectre Guide Documents*, Cadence, Berkshire, U.K., 2008. [Online]. Available: http://www.cadence.com.

[23] *Eldo Classic Integrated Circuit Simulation Datasheet*, Mentor Graphics Corporation, Wilsonville, OR, 2011. [Online]. Available: http://www.mentor.com.

[24] *Analog FastSPICE Platform Datasheet*, Berkeley Design Automation Inc., Santa Clara, CA, 2010. [Online]. Available: http://www.berkeley-da.com.

[25] K. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *Antennas and Propagation, IEEE Transactions on*, vol. 14, no. 3, pp. 302–307, May 1966.

[26] J. Schutt-Aine, "Stability analysis of the latency insertion method using a block matrix formulation," in *Electrical Design of Advanced Packaging and Systems (EDAPS), IEEE Symposium on*, Dec. 2008, pp. 155–158.

[27] J. P. Hespanha, *Linear Systems Theory*. Princeton, NJ: Princeton University Press, 2009.

[28] P. Goh, J. Schutt-Aine, D. Klokotov, J. Tan, P. Liu, W. Dai, and F. Al-Hawari, "Partitioned latency insertion method (PLIM) with stability considerations," in *Signal Propagation on Interconnects (SPI), IEEE 15th Workshop on*, May 2011, pp. 107–110.

[29] P. Goh, J. Schutt-Aine, D. Klokotov, J. Tan, P. Liu, W. Dai, and F. Al-Hawari, "Partitioned latency insertion method with a generalized stability criteria," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 1, no. 9, pp. 1447–1455, Sep. 2011.

[30] P. Liu, J. Tan, Z. Zhou, J. Schutt-Aine, and P. Goh, "A comparison of two latency insertion methods in dependent sources applications," in *Electrical Performance of Electronic Packaging and Systems (EPEPS), IEEE 20th Conference on*, Oct. 2011, pp. 295–298.

[31] P. Liu, J. Tan, Z. Zhou, J. Schutt-Aine, and P. Goh, "Application of the amplification matrix latency insertion method to circuits with dependent sources," in *Electrical Design of Advanced Packaging and Systems (EDAPS), IEEE Symposium on*, Dec. 2011.

[32] R. Gao and J. Schutt-Aine, "Improved latency insertion method for simulation of large networks with low latency," in *Electrical Performance of Electronic Packaging (EPEP), IEEE 11th Topical Meeting on*, 2002, pp. 37–40.

[33] H. Asai and N. Tsuboi, "Multi-rate latency insertion method with RLCG-MNA formulation for fast transient simulation of large-scale interconnect and plane networks," in *Electronic Components and Technology (ECTC), IEEE 57th Conference on*, May-Jun. 2007, pp. 1667–1672.

[34] N. Tsuboi and H. Asai, "Multi-rate latency insertion method for the fast transient simulation of large networks with nonlinear termination," in *Electrical Performance of Electronic Packaging (EPEP), IEEE 15th Topical Meeting on*, Oct. 2006, pp. 137–140.

[35] Z. Deng and J. Schutt-Aine, "Stability analysis of latency insertion method (LIM)," in *Electrical Performance of Electronic Packaging (EPEP), IEEE 13th Topical Meeting on*, Oct. 2004, pp. 167–170.

[36] S. Lalgudi, M. Swaminathan, and Y. Kretchmer, "On-chip power-grid simulation using latency insertion method," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 55, no. 3, pp. 914–931, Apr. 2008.

[37] S. Lalgudi and M. Swaminathan, "Analytical stability condition of the latency insertion method for nonuniform GLC circuits," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 55, no. 9, pp. 937–941, Sep. 2008.

[38] J. Schutt-Aine, J. Tan, C. Kumar, and F. Al-Hawari, "Blackbox macromodel with S-parameters and fast convolution," in *Signal Propagation on Interconnects (SPI), IEEE 12th Workshop on*, May 2008, pp. 1–4.

[39] M. R. Wohlers, *Lumped and Distributed Passive Networks*. New York, NY: Academic, 1969.

[40] S. Grivet-Talocia and A. Ubolli, "On the generation of large passive macromodels for complex interconnect structures," *Advanced Packaging, IEEE Transactions on*, vol. 29, no. 1, pp. 39–54, Feb. 2006.

[41] S. Boyd, L. El Ghaoui, E. Feron and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. Hoboken, NJ: Wiley, SIAM Studies in Applied Mathematics, vol. 15, 1994.

[42] D. Saraswat, R. Achar, and M. Nakhla, "Fast passivity verification and enforcement via reciprocal systems for interconnects with large order macromodels," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 1, pp. 48–59, Jan. 2007.

[43] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*. Boston, MA: Academic, 1990.

[44] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. New York, NY: Cambridge University Press, 1996.

[45] K. Zhou and D. J. C. Doyle, *Essentials of Robust Control*. Upper Saddle River, NJ: Prentice Hall, 1998.

[46] A. Semlyen and A. Dabuleanu, "Fast and accurate switching transient calculations on transmission lines with ground return using recursive convolutions," *Power Apparatus and Systems, IEEE Transactions on*, vol. 94, no. 2, pp. 561–571, Mar. 1975.

[47] J. Schutt-Aine, J. Tan, C. Kumar, "Use of Smith chart to compensate for missing data on network performance at lower frequency," patent application, Oct. 2007.

[48] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 1999.

[49] J. Schutt-Aine, P. Goh, Y. Mekonnen, J. Tan, F. Al-Hawari, P. Liu, and W. Dai, "Comparative study of convolution and order reduction techniques for blackbox macromodeling using scattering parameters," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 1, no. 10, pp. 1642–1650, Oct. 2011.

[50] J. Schutt-Aine, D. Klokotov, P. Goh, J. Tan, F. Al-Hawari, P. Liu, and W. Dai, "Application of the latency insertion method to circuits with blackbox macromodel representation," in *Electronics Packaging Technology (EPTC), IEEE 11th Conference on*, Dec. 2009, pp. 92–95.

[51] *Agilent Advanced Design System (ADS)*, Agilent Technologies, Santa Clara, CA, 2009. [Online]. Available: http://www.agilent.com.

[52] J. Choi, M. Swaminathan, N. Do, and R. Master, "Modeling of power supply noise in large chips using the circuit-based finite-difference time-domain method," *Electromagnetic Compatibility, IEEE Transactions on*, vol. 47, no. 3, pp. 424–439, Aug. 2005.

[53] T. Sekine and H. Asai, "CMOS circuit simulation using latency insertion method," in *Electrical Performance of Electronic Packaging (EPEP), IEEE 17th Conference on*, Oct. 2008, pp. 55–58.

[54] D. Klokotov, P. Goh, and J. Schutt-Aine, "Latency insertion method (LIM) for DC analysis of power supply networks," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 1, no. 11, pp. 1839–1845, Nov. 2011.

[55] P. Goh and J. E. Schutt-Aine, "Latency insertion method (LIM) for CMOS circuit simulations with multi-rate considerations," in *Electrical Performance of Electronic Packaging and Systems (EPEPS), IEEE 20th Conference on*, Oct. 2011, pp. 125–128.

[56] M. Perrott, "Fast and accurate behavioral simulation of fractional-N frequency synthesizers and other PLL/DLL circuits," in *ACM/EDAC/IEEE 39th Design Automation Conference (DAC)*, 2002, pp. 498–503.

[57] S. Sancho, A. Suarez, and J. Chuan, "General envelope-transient formulation of phase-locked loops using three time scales," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 52, no. 4, pp. 1310–1320, Apr. 2004.

[58] S. Goldman, *Phase-Locked Loops Engineering Handbook for Integrated Circuits*. Norwood, MA: Artech House, 2007.

[59] G. Bianchi, *Phase-Locked Loop Synthesizer Simulation*. New York, NY: McGraw-Hill, 2005.

[60] A. Kral, F. Behbahani, and A. Abidi, "RF-CMOS oscillators with switched tuning," in *Custom Integrated Circuits Conference, Proceedings of the IEEE 1998*, May 1998, pp. 555–558.

[61] W. Sui, D. Christensen, and C. Durney, "Extending the two-dimensional FDTD method to hybrid electromagnetic systems with active and passive lumped elements," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 40, no. 4, pp. 724–730, Apr. 1992.

[62] M. Piket-May, A. Taflove, and J. Baron, "FD-TD modeling of digital signal propagation in 3-D circuits with passive and active loads," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 42, no. 8, pp. 1514–1523, Aug. 1994.

[63] V. Thomas, M. Jones, M. Piket-May, A. Taflove, and E. Harrigan, "The use of SPICE lumped circuits as sub-grid models for FDTD analysis," *Microwave and Guided Wave Letters, IEEE*, vol. 4, no. 5, pp. 141–143, May 1994.

[64] P. Ciampolini, P. Mezzanotte, L. Roselli, and R. Sorrentino, "Accurate and efficient circuit simulation with lumped-element FDTD technique," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 44, no. 12, pp. 2207–2215, Dec. 1996.

[65] C. Kuo, B. Houshmand, and T. Itoh, "Full-wave analysis of packaged microwave circuits with active and nonlinear devices: An FDTD approach," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 45, no. 5, pp. 819–826, May 1997.

[66] Z. Deng and J. Schutt-Aine, "LIM-SPICE for the analysis of power distribution networks," in *Signal Propagation on Interconnects (SPI), IEEE 9th Workshop on*, May 2005, pp. 17–20.

[67] Z. Deng and J. Schutt-Aine, "Turbo-SPICE with latency insertion method (LIM)," in *Electrical Performance of Electronic Packaging (EPEP), IEEE 14th Topical Meeting on*, Oct. 2005, pp. 329–332.